Advant® OCS
with Master software

Advant® Station 500 Series Engineering Station

User’s Guide

Use of **DANGER**, **WARNING**, **CAUTION**, and **NOTE**

This publication includes, **DANGER**, **WARNING**, **CAUTION**, and **NOTE** information where appropriate to point out safety related or other important information.

DANGER	Hazards which could result in severe personal injury or death
WARNING	Hazards which could result in personal injury
CAUTION	Hazards which could result in equipment or property damage
NOTE	Alerts user to pertinent facts and conditions

Although **DANGER** and **WARNING** hazards are related to personal injury, and **CAUTION** hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, comply fully with all **DANGER**, **WARNING**, and **CAUTION** notices.

TRADEMARKS

Advant, AdvaCommand, AdvaInform, and AdvaBuild are registered trademarks of ABB Asea Brown Boveri Ltd, Switzerland.

AutoCAD and DXF are registered trademarks of Autodesk Inc.

FrameMaker is a trademark of Frame Technology Corp.

HP and all HP products described in this document are registered trademarks of Hewlett Packard Company

IBM-PC is a trademark of IBM Corporation

MOD 300 is a trademark of ABB Process Automation Inc.

ORACLE and SQL*ReportWriter are registered trademarks of Oracle Corporation

OSF/Motif is a trademark of The Open Software Foundation, Inc.

UNIX is a trademark of AT&T

NOTICE

The information in this document is subject to change without notice and should not be construed as a commitment by <ABB Company>. <ABB Company> assumes no responsibility for any errors that may appear in this document.

In no event shall <ABB Company> be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall <ABB Company> be liable for incidental or consequential damages arising from use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without <ABB Company>’s written permission, and the contents thereof must not be imparted to a third party nor be used for any unauthorized purpose.

The software described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license.

CE MARKING

This product meets the requirements specified in EMC Directive 89/336/EEC and in Low Voltage Directive 72/23/EEC.

Copyright © <ABB Company> 1997

3BSE001264/D

Template: 3BSE001286/E

TABLE OF CONTENTS

Chapter 1 - Introduction

1.1 General Information 1-1
1.2 Equipment Requirements 1-1
1.3 Manual Organization 1-2
1.4 Conventions 1-3
1.5 Related Documentation 1-4
1.6 Release History 1-5
1.7 Terminology 1-7
1.8 Product Overview 1-8
  1.8.1 AdvaBuild Basic Functions 1-9
    1.8.1.1 Structure Builder 1-9
    1.8.1.2 Template Builder 1-18
    1.8.1.3 AdvaBuild Function Chart Builder 1-20
    1.8.1.4 SQL*ReportWriter Interface 1-22
    1.8.1.5 Plant Documentation Builder 1-22
  1.8.2 AdvaBuild On-line Builder 1-22
  1.8.3 Import/Export Utility 1-23
  1.8.4 Optional Builders 1-23
    1.8.4.1 AdvaBuild Diagram Builder 1-23
    1.8.4.2 AdvaBuild Electrical Diagram Builder 1-24
    1.8.4.3 AdvaBuild Display Builder 1-26
    1.8.4.4 AdvaBuild Document Builder 1-28
1.9 User Interface for AdvaBuild Basic Software 1-29
  1.9.1 General User Interface 1-29
  1.9.2 Using a Mouse 1-29
  1.9.3 Working with Windows 1-30
    1.9.3.1 Using a Dialog Box 1-33
    1.9.3.2 Menus 1-34
  1.9.4 FORMS User Interface 1-34
    1.9.4.1 Function Keys 1-35
    1.9.4.2 Special Key Functions 1-36
    1.9.4.3 Status Line 1-37
    1.9.4.4 Message Line 1-37
    1.9.4.5 Data Area and Dialog Boxes 1-38
    1.9.4.6 Menus 1-38

CONTENTS (continued)

Chapter 2 - Installation

2.1 Site Planning/Environment 2-1
2.2 Setup 2-1
2.3 Power Up Procedure 2-1
2.4 Start-up Procedure 2-1
2.5 Shut-down Procedure 2-2
  2.5.1 Power Down the Engineering Station 2-2

Chapter 3 - Configuration/Application Building

3.1 Application Start-up 3-1
  3.1.1 Log In To HP-UX 3-1
    3.1.1.1 Changes in the HP-VUE environment compared with the HP-VUE standard 3-2
  3.1.2 Logging In to a Specific Project or the Administration Function 3-4
3.2 Tutorial Master Applications 3-5
  3.2.1 Overview 3-5
  3.2.2 Logging In 3-6
  3.2.3 Navigating the Project Structure 3-7
  3.2.4 Using the Template Builder 3-9
    3.2.4.1 Opening an Object and Invoking the Template Builder ... 3-9
    3.2.4.2 Editing Object Attributes 3-10
    3.2.4.3 Saving and Exiting an Object 3-11
  3.2.5 Inserting an Object 3-12
  3.2.6 Inserting Objects via the Template Builder Multiple-Object Form 3-13
  3.2.7 Editing Multiple Objects from the Multiple-object Form 3-17
  3.2.8 Moving Objects from the List to the Tree Presentation 3-18
  3.2.9 Returning to the Project Root 3-20
  3.2.10 Printing a Documentation Package 3-20
3.3 On-line Builder Tutorial 3-24
  3.3.1 Overview 3-24
  3.3.2 Preparations 3-25
  3.3.3 Using the On-line Builder 3-41
3.4 Application Procedures 3-52
  3.4.1 Application Procedures For Administration 3-52
    3.4.1.1 Administration Functions Overview 3-52
    3.4.1.2 How to Start the Administration Function 3-60
    3.4.1.3 How to Create a New Project 3-60
    3.4.1.4 How to Insert a User 3-63

CONTENTS (continued)

3.4.1.5 How to Save a Project Release 3-65
3.4.1.6 How to Restore a Project Release 3-67
3.4.1.7 How to Modify a Project's Comment and Protection.... 3-69
3.4.1.8 How to Enlarge a Project 3-70
3.4.1.9 How to Delete a Project 3-71
3.4.1.10 How to Handle a Corrupted Project............................... 3-72
3.4.1.11 How to Modify a User's Password 3-74
3.4.1.12 How to Delete a User 3-75
3.4.2 How to Build the Project Structure via the Structure Builder 3-76
3.4.2.1 How to Open a Project in the Structure Builder 3-77
3.4.2.2 How to Navigate in the Tree Presentation 3-78
3.4.2.3 How to Access the List Presentation from the Tree Presentation.. 3-79
3.4.2.4 How to Navigate in the List Presentation 3-80
3.4.2.5 How to Create New Objects in the Project Structure..... 3-81
3.4.2.6 How to Move, Delete, and Rename Objects.................. 3-82
3.4.2.7 Help Window 3-82
3.4.2.8 Error Messages... 3-82
3.4.2.9 Object Information Window 3-83
3.4.3 How to use the Template Builder 3-84
3.4.3.1 How to Start the Template Builder 3-84
3.4.3.2 How to Save Your Work in the Template Builder......... 3-87
3.4.3.3 How to Edit Object Attributes via the Template Builder.. 3-88
3.4.3.4 How to Insert Objects via the Template Builder............ 3-89
3.4.4 Application Procedures for Link Handling.. 3-91
3.4.4.1 Link Handling Windows.. 3-92
3.4.4.2 How to Start the Link Handling Function...................... 3-92
3.4.4.3 Link Type Specific Window 3-93
3.4.4.4 General Link Handling Window.................................... 3-95
3.4.4.5 How to Create Links 3-96
3.4.4.6 How to Query for Links 3-97
3.4.5 How to Print Plant Documentation... 3-99
3.4.5.1 Plant Documentation Builder Overview 3-99
3.4.5.2 Document Types 3-104
3.4.5.3 How to Configure the DOC_RELATION Table 3-108
3.4.5.4 How to Build the Documentation Structure................. 3-112
3.4.5.5 How to Define the Object Attributes for a Documentation Package 3-112

CONTENTS (continued)

3.4.5.6 How to Create Links for DSTRUC Objects 3-113
3.4.5.7 How to Generate PDB Packages 3-114
3.4.5.8 How to Print Document Packages 3-115
3.4.6 Configure Reports (Application Procedure for SQL*ReportWriter).. 3-120
3.4.6.1 Overview SQL*ReportWriter and AdvaBuild Integration 3-120
3.4.6.2 How to Create and Modify Reports 3-123
3.4.6.3 How to Define a Report 3-130
3.4.6.4 How to Test the Report 3-138
3.4.6.5 Terminate Session 3-139
3.4.6.6 How to Use the Report in the Plant Documentation Builder 3-140
3.4.6.7 Execute a Report 3-140
3.4.6.8 Standard Footer 3-141
3.4.7 Application Procedures for Solutions 3-143
3.4.7.1 How to Use Solutions 3-143
3.4.8 How to Integrate and Access the Display Builder 3-147
3.4.8.1 Handling of Display Objects with Structure Builder ... 3-147
3.4.8.2 How to Start the Display Builder Via the Structure Builder 3-150
3.4.8.3 Open a Display from Clipboard Directory 3-151
3.4.8.4 How to Create an Object List for insertion into DISPLAY BUILDER 3-152
3.4.8.5 How to Use the Distributor 3-156
3.4.8.6 Exit Display Builder 3-158
3.4.9 Export/Import Utility 3-159
3.4.9.1 Introduction 3-159
3.4.9.2 Basic Terms 3-160
3.4.9.3 Overview 3-160
3.4.9.4 Import 3-163
3.4.9.5 Export 3-172
3.4.9.6 **Examples** (based on the project MAS_EXP) 3-176
3.4.10 How to Import Existing FrameMaker and AutoCAD Files 3-180
3.4.10.1 Handling Floppy Disks 3-180
3.4.10.2 How to Import an Existing FrameMaker File 3-181
3.4.10.3 How to Import an Existing AutoCAD file 3-181
3.4.11 Queries 3-182
3.4.11.1 How to Enter a Query 3-182
3.4.11.2 Count Hits 3-183

CONTENTS (continued)

3.4.11.3 How to Execute a Query 3-184
3.4.11.4 Cancel 3-184
3.5 Menus 3-185
3.5.1 Common Menus 3-185
3.5.1.1 Query Menu 3-185
3.5.1.2 Help Menu 3-187
3.5.2 Menus for the Administration Window 3-188
3.5.2.1 Administration Window - File Menu 3-188
3.5.2.2 Administration Window - Administer Menu 3-189
3.5.3 Menus for Projects Window 3-189
3.5.3.1 Projects Window - File Menu 3-191
3.5.3.2 Projects Window - Edit Menu 3-194
3.5.4 Menus for Users Window 3-197
3.5.4.1 Users Window - File Menu 3-198
3.5.4.2 Users Window - Edit Menu 3-198
3.5.5 Menus for Structure Builder Tree and List Presentation Windows 3-201
3.5.5.1 File Menu 3-201
3.5.5.2 Edit Menu 3-205
3.5.5.3 View Menu 3-211
3.5.5.4 Object Menu 3-216
3.5.5.5 Link Menu 3-220
3.5.5.6 Query Menu 3-223
3.5.5.7 Utility Menu 3-224
3.5.5.8 Help Menu 3-226
3.5.6 Menus for Structure Builder Object Information 3-227
3.5.6.1 File Menu 3-227
3.5.6.2 View Menu 3-228
3.5.6.3 Query Menu 3-228
3.5.7 Menus for Solution Aspects Window 3-229
3.5.8 Menus for Plant Documentation Builder Configure Window 3-233
3.5.8.1 File Menu 3-233
3.5.8.2 Edit Menu 3-235
3.5.8.3 View Menu 3-237
3.5.8.4 Query Menu 3-237
3.5.8.5 Help Menu 3-237
3.5.8.6 Special Keys 3-237

CONTENTS (continued)

3.5.9 Menus for the Plant Documentation Builder Generation Window 3-238

3.5.9.1 File Menu 3-238

3.5.9.2 View Menu 3-239

3.5.9.3 Document Menu 3-240

3.5.9.4 Query Menu 3-241

3.5.9.5 Help Menu 3-241

3.5.10 Link Handler Menus 3-241

3.5.10.1 File Menu 3-241

3.5.10.2 Edit Menu 3-244

3.5.10.3 View Menu 3-246

3.5.10.4 Object Menu 3-248

3.5.10.5 Link Menu 3-249

3.5.10.6 Query Menu 3-250

3.5.10.7 Special Key Functions 3-250

3.5.11 Menus for the Template Builder 3-251

3.5.11.1 File Menu 3-251

3.5.11.2 Edit Menu 3-255

3.5.11.3 View Menu 3-259

3.5.11.4 Link Menu 3-261

3.5.11.5 Query Menu 3-262

3.5.11.6 Special Key Functions 3-262

3.6 Data Exchange with Advant Station 100 Series Stations 3-263

3.6.1 Introduction 3-263

3.6.2 How to exchange Data via Floppy Disk 3-266

3.6.3 How to establish an On-line Connection via Network 3-268

3.6.4 Import of AS 100ES Files 3-269

3.6.5 Export of AS 100ES Files 3-275

3.6.6 Details about export/import 3-278

3.6.6.1 AS 100ES Directory Structure on the AS 500ES 3-278

3.6.6.2 Automatic Conversion during Export/Import 3-280

3.6.6.3 Constraints and Dependencies 3-281

3.7 On-line Builder 3-281

3.7.1 Introduction 3-281

3.7.2 Preparation 3-282

3.7.3 How to Boot the RealTimeAccelerator 3-282

3.7.4 How to Start the On-line Builder 3-283

3.7.5 On-line Builder Convert Dialog window 3-283

3.7.6 How to Prepare Source Code for On-line Builder 3-286

CONTENTS (continued)

- 3.7.7 How to Work with the On-line Builder 3-287
- 3.7.8 How to Extract PC Source Code 3-288
- 3.7.9 How to Change the Configuration of Net, Node 3-289
- 3.7.10 How to Change the Configuration of Printers 3-291
- 3.7.11 How to Use the On-line Builder with a MP51 3-293
- 3.7.12 Central Backup and Restore 3-293
  - 3.7.12.1 How to Backup a Complete Plant 3-293
  - 3.7.12.2 How to Restore a Complete Plant 3-294

3.8 How to Edit PC and DB Source Code 3-294

Chapter 4 - Runtime Operation

4.1 Product Operation 4-1

4.2 Operating Overview 4-1

4.3 Runtime Tutorial 4-1

4.4 Operating Instructions 4-1

4.5 Runtime Operation Menus 4-1

Chapter 5 - Maintenance

5.1 Error Messages 5-1

- 5.1.1 System Administration Error Messages 5-1
  - 5.1.1.1 Location of Error Messages 5-1
  - 5.1.1.2 General Error Messages 5-1
  - 5.1.1.3 Error Messages Caused by Concurrent Execution of Administrative Tasks 5-2
  - 5.1.1.4 Error Messages Caused by 'Insert / Delete Project' 5-3
  - 5.1.1.5 Error Messages Caused by 'Enlarge Project' 5-4
  - 5.1.1.6 Error Messages Caused by 'Save Project' 5-5
  - 5.1.1.7 Error Messages Caused by 'Restore Project' 5-5
  - 5.1.1.8 Error Messages Caused by Disc Space Check during Insert/Save/Enlarge Project 5-6
  - 5.1.1.9 Error Messages Caused by 'Insert / Delete User' 5-7
- 5.1.2 Structure Builder Error Messages 5-7
- 5.1.3 Template Builder Error Messages 5-17
- 5.1.4 From AdvaBuild Structure Builder Object Information 5-22
- 5.1.5 Link Handling Messages 5-24
- 5.1.6 Plant Documentation Builder Error Messages 5-30
- 5.1.7 Export/Import Error Messages 5-35
- 5.1.8 From AdvaBuild On-line Builder 5-40

5.2 Export Import Utility Restrictions 5-45

CONTENTS (continued)

Appendix A - AS500ES Directory Structure

A.1 Top Directory, Principle Scheme A-1
A.2 Products Structure A-1
A.3 Data Directory Structure A-4
A.4 The proj Directory Structure A-5
A.5 The proj_adm Directory Structure A-6
A.6 The <project> Directory Structure A-7
A.7 AS500ES Environment Variables A-10
A.8 Environment Variables Referring to Products Structure A-10
A.9 Environment Variables Referring to Data Directory Structure A-11
A.10 Environment Variables Referring to proj_adm Directory Structure A-12
A.11 Environment Variables Referring to proj Directory Structure A-12
A.12 Environment Variables Referring to User Specific Directories A-13
A.13 General Environment Variables Setting A-13

Appendix B - Glossary

B.1 Abbreviations B-1
B.2 Terms B-5

Appendix C - Guideline to combine EIU with HP/UX Shellscript Programming

C.1 Introduction C-1
C.2 Background and Setup C-1
C.3 Example of inserting a single object C-2
C.4 Example of inserting a number of objects C-3
  C.4.1 Initializing the files C-4
  C.4.2 Inserting objects into files C-4
  C.4.3 Inserting objects into database C-4
  C.4.4 Code of shellscript obj C-5
C.5 Using nested shellscript files C-6
  C.5.1 Example for a TAG typical C-6
  C.5.2 Example for multiple board element creation C-7
C.6 Example for automatic channel assignment C-7
  C.6.1 Reading free signal elements from the database C-8
  C.6.2 Reading free channels from the database C-8
  C.6.3 Assigning signals to free channels C-9
  C.6.4 Importing of the channel assignment C-9
  C.6.5 Code of shellscript i_o C-9

CONTENTS (continued)

Appendix D - Examples for EIU

D.1 EIU Export D-1
  D.1.1 Simple Export with TEB of some Data from one Table D-1
  D.1.2 Simple Export with the Command Line of some Data from one Table . D-2
  D.1.3 Export of Data from Different Tables with Select File D-2
  D.1.4 Extract Files (Texts) from DB with EIU D-3
D.2 EIU Import D-4
  D.2.1 Import Files (Drawings) in DB with EIU D-4
  D.2.2 Import of Digital Signals with Structure Objects, Boards and Links D-6

Appendix E - Use of Microsoft Word 6.0 documents in AS500ES

E.1 Introduction E-1
E.2 How to establish Wabi, Microsoft Windows and Microsoft Word E-2
  E.2.1 Installation of Software for Use of Microsoft Word E-2
  E.2.2 Software Configuration for Use in AS500ES E-2
  E.2.3 Check Correct Installation and Configuration E-3
  E.2.4 Microsoft Word Configuration for Communication with STB E-3
  E.2.5 Creating of Microsoft Word Document Directory E-3
E.3 Open, Print in STB from Microsoft Word Documents E-4
  E.3.1 Creation of FREE Object for Microsoft Word E-4
  E.3.2 STB-Open of Microsoft Word Document with FREE Object as Representatives E-4
  E.3.3 STB-Print of Microsoft Word Document with FREE Object as Representative E-5
  E.3.4 PDB Configuration for Microsoft Word 6.0 Documents E-5
  E.3.5 Import of Microsoft Word 6.0 Documents from other Systems E-6

Appendix F - Use of Time controlled Central Backup with Free Object

F.1 Introduction F-1
  F.1.1 Time controlled Central Backup Function in MAS_EXP Project F-1
    F.1.1.1 Stop of an already running Time controlled Central Backup F-2
    F.1.1.2 Start of a Time controlled Central Backup F-2
    F.1.1.3 Restore of Time controlled Central Backup files F-4
  F.1.2 Installation of Time controlled Central Backup in other Projects F-5

CONTENTS (continued)

F.1.3 Working Method of Time controlled Central Backup F-5
  F.1.3.1 STOP_CRON F-5
  F.1.3.2 START_CRON F-5
F.1.4 Appendix for Time controlled Central Backup....................................... F-6
  F.1.4.1 Manual Page of crontab.. F-6
  F.1.4.2 Manual Page of cron... F-7

INDEX

Chapter 1 Introduction

1.1 General Information

AdvaBuild basic software are a set of software packages that support an object-oriented approach for database configuration and documentation. This set is composed of the AdvaBuild Structure Builder, Template Builder, Function Chart Builder, and Plant Documentation Builder. In addition to these basic functions, optional AdvaBuild software packages are available for other system engineering functions such as On-line Builder, Diagram Builder, Electrical Diagram Builder, Display Builder, and so on. Instructions for these optional AdvaBuild packages are provided in separate instruction books as described in Section 1.5, Related Documentation.

This book provides instructions for operating the Structure Builder, Template Builder, and Plant Documentation Builder. This book is intended to be used in conjunction with *AdvaBuild Engineering Methods for the Master System*. Before you actually use these builders, you should consult the *Engineering Methods* book to become familiar with the principles of object-oriented engineering, and for database planning guidelines.

The *Advant Station 500 Engineering Station User's Guide* and *AdvaBuild Engineering Methods* books are not intended to be the sole source of instruction for AdvaBuild software. It is recommended that those people involved in system engineering attend the applicable system engineering courses offered by ABB Industrial Systems, Inc.

1.2 Equipment Requirements

AdvaBuild Basic software run on the Advant Station 500 Series Engineering Station.
For Demo- and Training equipment, the AS 500ES software can be installed on a separate external disk to an Operator Station. So the workstation can be booted either as an Operator Station or an Engineering Station.

The Advant Station comes equipped with the hardware and software required to support the software packages that you order. The memory provided to support AdvaBuild Basic Functions, and the memory requirements for various options are described in the *Advant Station 515/520i Hardware User's Guide*

1.3 Manual Organization

This book is organized into chapters and sections as shown in Figure 1-1.

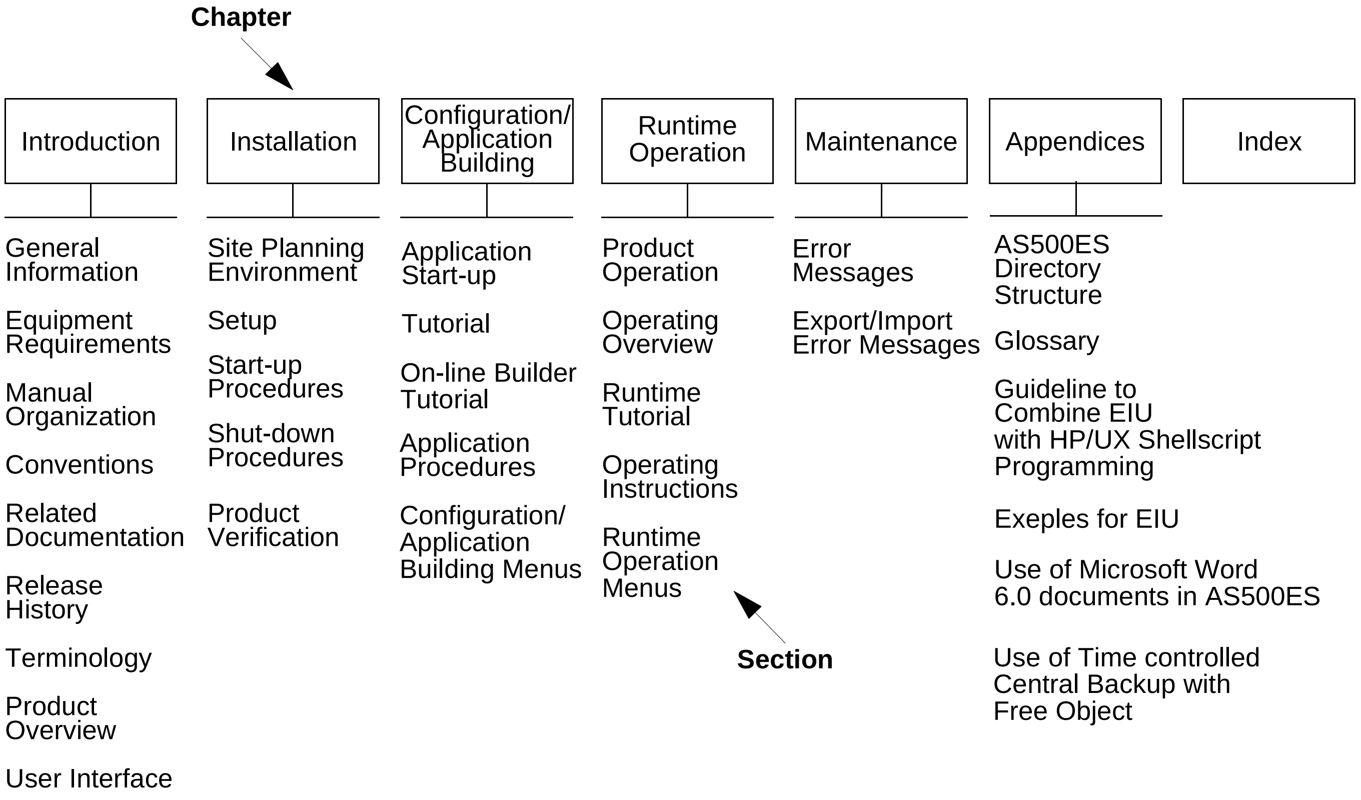


Figure 1-1. Manual Organization

How to Use this Book

This book is organized to facilitate learning AdvaBuild basic functions without having to read this book from cover to cover. First, refer to {Ref. Master Tutorial} to get some quick “hands-on” experience. The tutorial is intended to familiarize you with the basic methods and concepts as quickly as possible, without getting bogged down in details. When you are finished with the tutorial, you should have a good understanding of the operation of most functions (even ones that are not covered in the tutorial).

If you need further information regarding a procedure that is not covered in the tutorial, refer to Section 3.4, Application Procedures. This section provides step-by-step instructions for basic configuration/application building procedures. Finally, if you need detailed information regarding a specific menu item used in an application procedure, refer to Section 3.5, Menus.

In the documentation structure, Chapter 4, Runtime Operation provides instructions and reference information related to runtime operation. The AdvaBuild basic functions described in this book are used exclusively for database configuration and documentation. There is no runtime component. Therefore this chapter is not applicable.

1.4 Conventions

The following conventions are used throughout this manual for the presentation of material:

- The words in proper names of screen elements (for example, the title in the title bar of a window, the label for a field of a dialog box) are initially capitalized.
- The names of keyboard keys are boldfaced.
  - Initial capital letters are used for the name of a keyboard key if it is labeled on the keyboard. For example, the **Return** key or **<Return>**.
  - Lowercase letters are used for the name of a keyboard key that is not labeled on the keyboard. For example, the **space bar**, **comma key**, and so on.
  - In descriptions, the word "key" is used in conjunction with its name. For example, pressing the **Return** key initiates the process.
  - In tables and procedures, angle brackets replace the word "key." For example, press **<Return>**.
  - A dash is used to indicate that you must simultaneously press several keys. For example, **<CTRL-C>** indicates that you must hold down the **CTRL** key while pressing the **C** key.
  - The phrase "press and release" is used to indicate that you sequentially press several keys. For example, to count the hits in a Template Builder, press and release **<ESC>**, and then **<C>**. This indicates that you press and release the **ESC** key, and then you press and release the **C** key.
- The names of push and toggle buttons are boldfaced.
  - Initial capital letters are used for the name of a push or toggle button labeled on the display. For example, the **OK** button.
  - In descriptions, tables and procedures, the word "button" is used in conjunction with its name. For example, pressing the **OK** button initiates the process.
- The names of menus and menu items are boldfaced.
  - Initial capital letters are used for the name of menus listed in the menu bar. For example, the **File** menu.
  - Initial capital letters are used for the name of menu items. For example, to open a new window, choose **New** from the **File** menu.
- System prompts/messages are shown in the Courier font, and user responses/inputs are in boldfaced Courier font. For example, if you enter a value out of range the following message is displayed:

  ```
  Entered value is not valid. The value must be 0-30.
  ```

  In a tutorial, you may be told to enter the string TIC132 in a field. The string is shown as follows in the procedure:

  `TIC132`
- 3BSE004501Rnn01 AdvaInform Report Builder User's Guide
  User's Guide for building Reports with SQL*ReportWriter

1.5 Related Documentation

A complete listing of AdvaBuild Documentation is shown in Figure 1-2 and is described below.

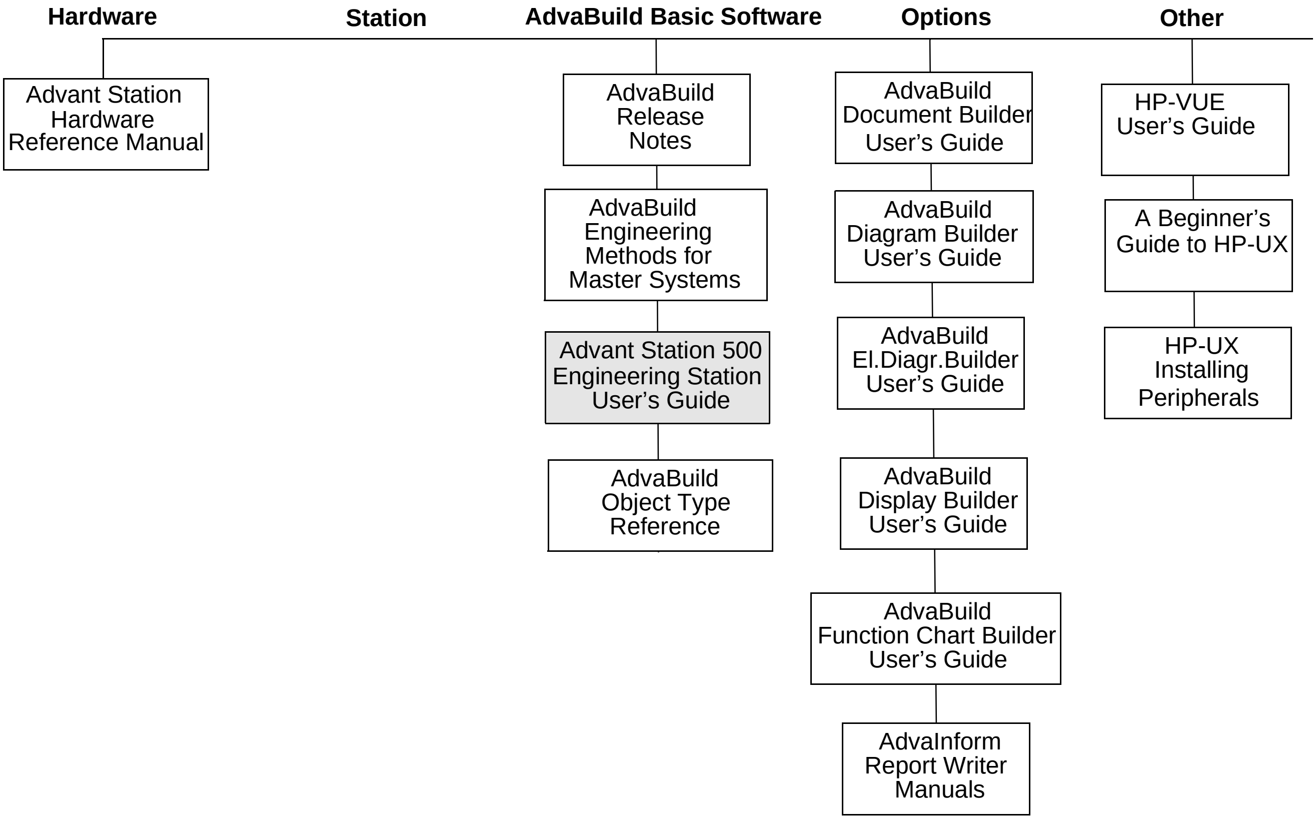

Hardware
Station
AdvaBuild Basic Software
Options
Other
Advant Station Hardware Reference Manual
AdvaBuild Release Notes
AdvaBuild Engineering Methods for Master Systems
Advant Station 500 Engineering Station User's Guide
AdvaBuild Object Type Reference
AdvaBuild Document Builder User's Guide
AdvaBuild Diagram Builder User's Guide
AdvaBuild El.Diagr.Builder User's Guide
AdvaBuild Display Builder User's Guide
AdvaBuild Function Chart Builder User's Guide
AdvaInform Report Writer Manuals
HP-VUE User's Guide
A Beginner's Guide to HP-UX
HP-UX Installing Peripherals

Figure 1-2. Related Documentation

- Advant Station 515 Hardware User's Guide or
  Advant Station 520i Hardware User's Guide

  Consult this book for instructions regarding hardware setup and installation.

- Release Notes

  Release notes are included in the delivery file with AdvaBuild software. The release notes provide installation instructions for both initial and upgrade installations, information regarding changes that have occurred since the last release of the project, and so on.

- AdvaBuild Engineering Methods for ABB Master Systems

  This book describes the principles of object-oriented engineering with AdvaBuild.

- Advant Station 500 Engineering Station User's Guide

  Consult this book for a general functional overview of the Engineering Station. This book also describes how to use the Structure Builder, Template Builder, Plant Documentation Builder, how to access Display Builder, Function Chart Builder and On-line Builder.

- AdvaBuild Object Type Reference Manual

  The Object Type Reference manual provides information on how to configure the various database object attributes.

- AdvaBuild Document Builder User's Guide
  This Binder is a documentation how to use the Document Builder, it includes template documentation as well.

- AdvaBuild Diagram Builder User's Guide
  This Binder is a description how to design drawings and diagrams with AutoCAD, including definition of user defined symbols. It includes library information as well.

- AdvaBuild Electrical Diagram Builder User's Guide

  This document contains a description of drawing support for function oriented electromechanical diagrams and layout drawings.

- AdvaBuild Display Builder User's Guide

- AdvaBuild Function Chart Builder User's Guide
  This book contains a description of AMPL application programs configuration.

- AdvaInform Reports Reference Manual
  Reference Manual for SQL*ReportWriter

- AdvaInform Report Builder User's Guide
  User's Guide for building Reports with SQL*ReportWriter

- AdvaBuild On-line Builder Engineering Stations User's Guide.

1.6 Release History

This book describes release 2.6 of the AdvaBuild Basic Functions. The following is a summary of AdvaBuild software releases up to and including release 2.6.

AdvaBuild Software Rel. 0

This release is for training and demonstration purposes. The platform for this release is the HP 9000/400 series Workstation. This product is supported until Rel. 0 Workstations are replaced by Rel. 1 Workstations.

AdvaBuild Software Rel. 1

This release is for ABB internal use with pilot projects. The platform: for this release is the HP 9000/720 Workstation. Enhancements provided by this release are as follows:

- Integrated Diagram Builder (described in *Diagram Builder User's Guide*)
- Integrated Function Chart Builder described in the Function Chart Builder User's Guide.

AdvaBuild Software Rel. 2.0

This is the first general release. The platform for this release is the HP 9000/720 and HP 9000/715 Workstations. This release provides On-line Builder functionality.

AdvaBuild Software Rel. 2.1

This release includes the following changes and enhancements:

- New Platform: Advant Station 520i
- Version for combined Demo- and Training Systems
- New Object Type: FREE object
- Integrated Display Builder
- Solution Handling
- Improved Diagram Builder based on AutoCAD12
- Integrated Electrical Diagram Builder
- New FrameMaker4 Version in Document Builder
- User Interface improvement
- Improvements on Error Messages.

AdvaBuild Software Rel. 2.1/3

This release includes the following changes and enhancements:

- Improved Installation procedure
- New Display Builder version with improved distribution integrated
- FCB for Master supports the Advant controller
- Bug fixes and several improvements.

AdvaBuild Software Rel. 2.1/4

This release includes the following changes and enhancements:

- New Boot and Base tape.
- Support of new hard-disk drive modules.

AdvaBuild Software Rel. 2.6/0

This release includes the following changes and enhancements:

- New data base objects to support S800 I/O
- Function Chart Builder 4.2 (Node Mode) ported from AdvaBuild for Windows (1.2) / AS100ES
- Enhanced Import/Export functions
- UPS supervision
- Support for RAID disks
- New icons on the HP VUE front panel for start-up and power off
- Database upgraded to Oracle 7
- New FrameMaker 5 version in Document Builder.

1.7 Terminology

Object

An object is a logical representation of functions, plant equipment, a location, documentation, or software function in the project database. For instance:

- A PROJECT object represents a complete project
- a TSTRUC object represents an location in the plant
- a PI_DIA object represents a P&I diagram
- a PCPGM object represents a PC program for a ABB Master process station.

Root

The object of type PROJECT is the top level object in the project database. This object stores information for project administration such as user names, project size, and so on. Initially this object is the root. Most objects can be set as a temporary root to facilitate navigation in the project hierarchy.

Tree

The project database is organized as a hierarchy of objects that are related by parent-child relationships.

Subtree

Part of a tree, for example a subplant underneath a PROJECT.

Parent/Child

A child and its siblings are hierarchically under a parent object from a subtree.

1.8 Product Overview

AdvaBuild software is an extension of the Computer Aided Plant Engineering (CAPE) concept of integrating plant engineering CAD capabilities with tools for DCS database configuration. This software supports the following project engineering functions:

- Configuration of the control database
- Project Documentation (including diagrams such as P&IDs and function charts).

AdvaBuild software consists of a family of modules called Builders that are integrated in a framework as shown in Figure 1-3. The framework consists of the basic components: Common database, and the Structure Builder as main user interface. Other builders (for example, Template Builder, Function Chart Builder) are connected in the frame as required.

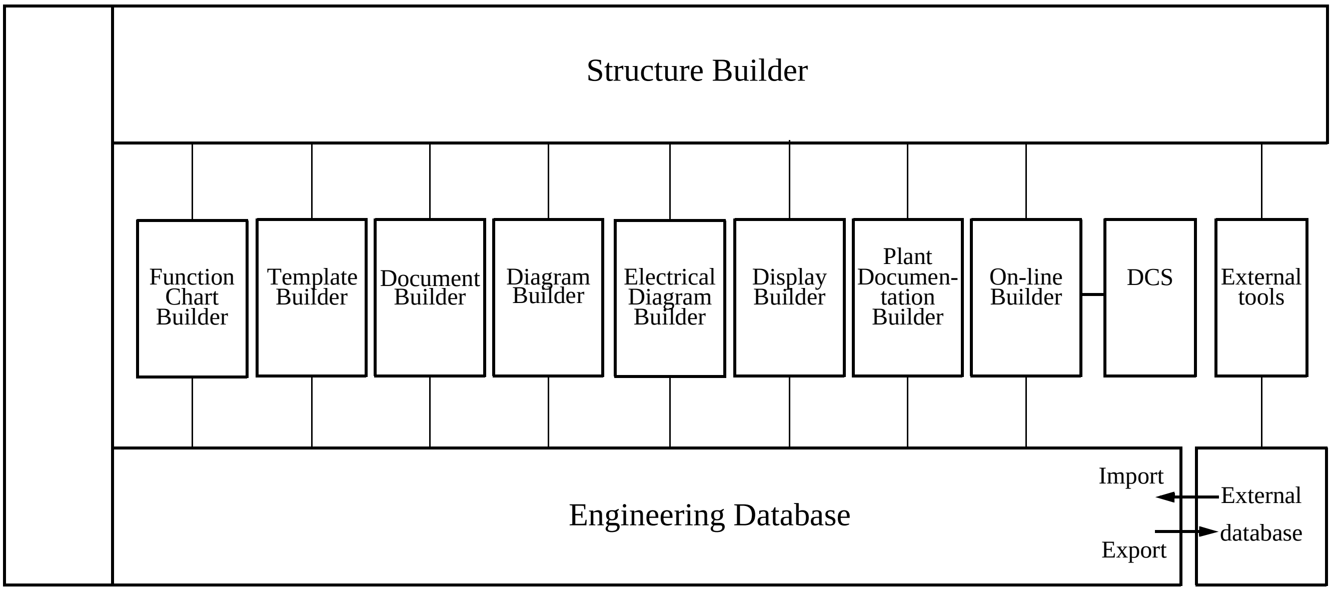


Figure 1-3. AdvaBuild Software Frame Architecture

These builders are highly integrated engineering tools that reduce engineering costs and delivery time, enhance the quality of project documentation, and relieve highly skilled persons from routine work. Since the builders are integrated in the framework, information that is input into one builder is automatically available and can be accessed by other builders. For instance, when you enter loop configuration data in the Template Builder, this data is also available to the Function Chart Builder for generating loop diagrams. Thus the data needs to be entered just once into the common database.

1.8.1 AdvaBuild Basic Functions

AdvaBuild basic functions include the Structure Builder, Template Builder, and Plant Documentation Builder.

1.8.1.1 Structure Builder

The Structure Builder integrates and provides access to all the other builders in the AdvaBuild framework. The Structure Builder also provides the means to build and maintain the object hierarchy for a project. This involves inserting, copying, moving, and deleting objects in the hierarchy. Thus, the Structure Builder is a convenient tool for managing a project, inserting new information, modifying existing information, and calling other AdvaBuild tools. The structure is constructed in a hierarchy according to the nature of the information objects.

The Structure Concept

The project consists of objects, as mentioned above. There are several types of objects: one for each control network, process stations, I/O card type, for PC programs, signals, diagrams, documentation, etc. These objects are tied together in a hierarchically, tree like project structure: PC programs may belong to process stations, process stations may belong to control networks, etc.

This project structure is represented by the Structure Builder as a graph of objects.

The basic purpose of the Structure Builder consists of defining functions-, location- and control system objects and inserting their corresponding object types at the appropriate positions in the plant structure. The Structure Builder also provides the ability to call other tools that facilitate inserting more detailed information into the objects. Instructions for using the functions provided by the Structure Builder are provided in the following sections.

Object Identification

Each object in the structure and in the whole project must be identified by a unique sequence of characters (an object identifier). These identifiers are used by the operations in the following sections. In principle there are no rules for defining object identifiers; however, it is recommended that you use the parent object identifier plus a sequence of characters for a child object identifier. Thus, if a parent object named “P” has two children, one with name “C1” and the other with name “C2”, the object identification of the children are “PC1” and “PC2”.

There are four main parts of the project structure. These are:

- The function oriented structure (ID “=”)
- The location oriented structure (ID “+”)
- The control system oriented structure (ID “#”)
- The plant documentation structure (ID “&”).

These structures each consist of objects in a sub-tree, describing the given aspect of the project. An example: AMPL programs are objects in the Control System Structure. The objects in the Location Oriented Structure represents the positioning of components in the plant area, and the functional descriptions reside in the Function Oriented Structure. Relations between these structures are documented using links. This means, for example, that a functional description can refer to corresponding application programs (AMPL) using links.

- Function Oriented Structure

  The function oriented structure describes how the system performs the defined operations with no regard to control nor location. It is a tree of objects that represents how functions are related, and documentation, such as diagrams etc. This structure provides a specification of the system, without information about the implementation of specific hardware or software.

- Location Oriented Structure

  This structure describes the physical locations of cards in racks, racks in cabinets, etc., up to locations in buildings or areas. It is a tree of objects that represents the locations that are relevant for positioning of devices, components, cabinets etc. This structure does not regard the function nor control of the system.

- Control System Oriented Structure

  The control oriented structure describes networks, nodes, application programs and their connection to signals. It is a tree of objects and their interrelations. This structure does not regard the physical location nor function of the system.

- Plant Documentation Structure

  This structure defines most of the documentation for the project. It is constructed in an orderly way, similar to the structures mentioned above. Objects in the plant documentation structure represent documentation packages and can be linked to objects in other structures using "`IS_DOCUMENTATION_OF`" links. These links define the scope of the documentation package.

The Structure Builder provides two modes for displaying the object structure. They are the Tree Presentation and the List Presentation.

Tree Presentation

The tree presentation is the default display mode. In this mode, objects are represented graphically as boxes, Figure 1-4. Navigation through the object structure in the Tree presentation is for example via the **View** menu, scroll bar, and embedded buttons in the tree structure.

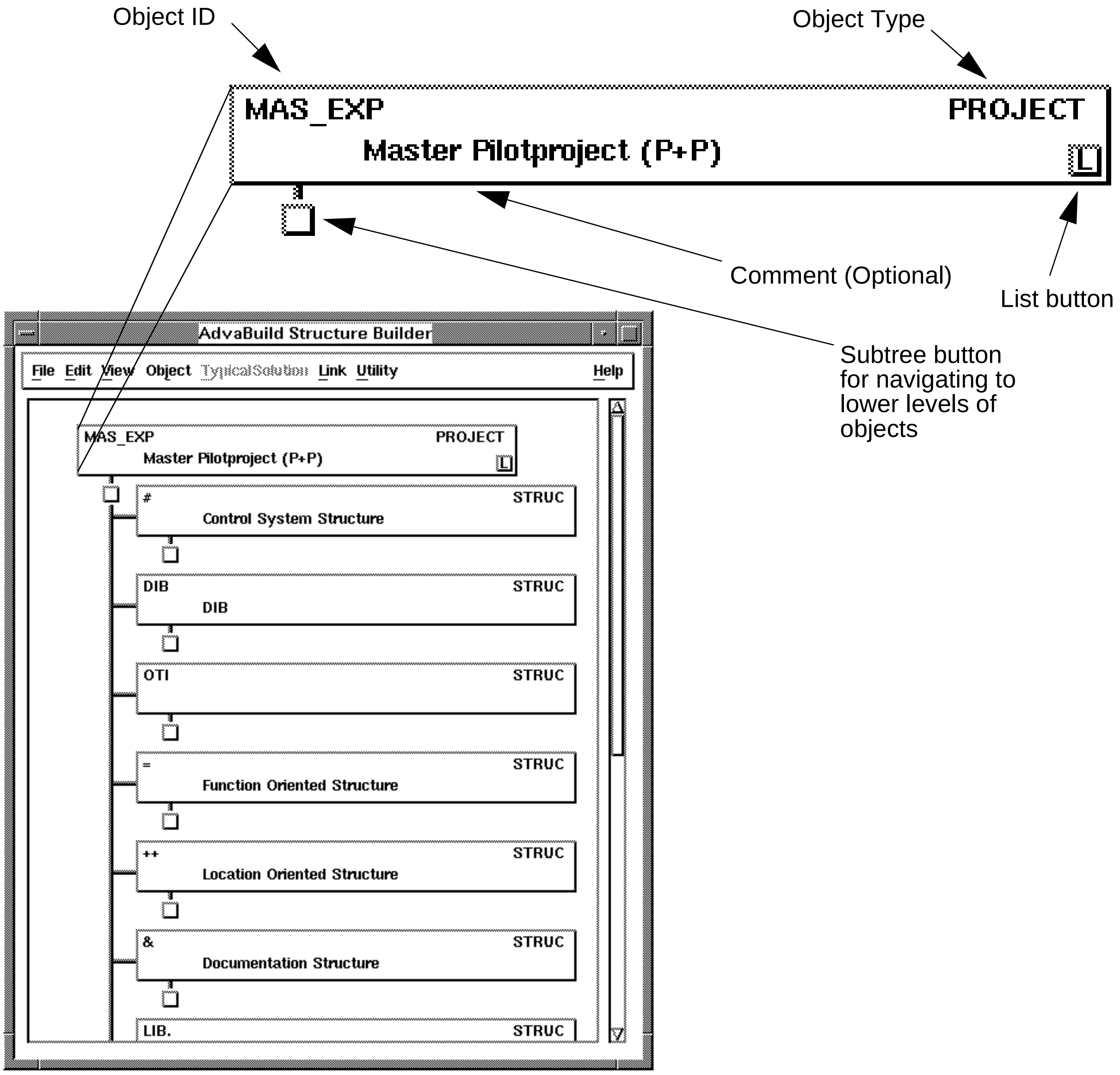
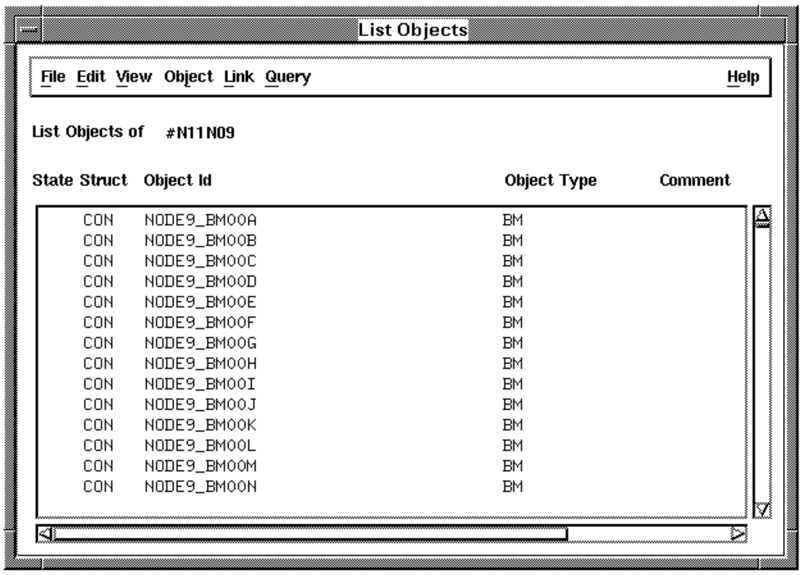


Figure 1-4. Structure Builder Tree Presentation

List Presentation

The list presentation, Figure 1-5, may be used as a means to input and view bulk data without cluttering and over complicating the tree presentation. The List Presentation is opened when you select the parent object, the **List** item from the **View** menu or click on the list button (shown above).



List Objects of #N11N09

State Struct	Object Id	Object Type	Comment
CON	NODE9_BM00A	BM	
CON	NODE9_BM00B	BM	
CON	NODE9_BM00C	BM	
CON	NODE9_BM00D	BM	
CON	NODE9_BM00E	BM	
CON	NODE9_BM00F	BM	
CON	NODE9_BM00G	BM	
CON	NODE9_BM00H	BM	
CON	NODE9_BM00I	BM	
CON	NODE9_BM00J	BM	
CON	NODE9_BM00K	BM	
CON	NODE9_BM00L	BM	
CON	NODE9_BM00M	BM	
CON	NODE9_BM00N	BM	

Figure 1-5. Structure Builder List Presentation

You can insert, copy, move, and otherwise manipulate objects in the list presentation as in the tree presentation. In the List presentation, you navigate either via the scroll bar or via queries.

Objects inserted into the tree presentation cannot be viewed in the list presentation. Likewise, objects inserted into the list presentation cannot be viewed in the tree presentation. However, you can move objects from one presentation to the other as required. That is tree objects can be removed from the tree presentation and placed in the list presentation. Likewise, list objects can be removed from the list presentation and placed in the tree presentation.

Table 1-1. AdvaBuild Object Types

CLASSIFICATION	**OBJECT-TYPE**	**Assignment to Structure-types**					
		=	**++/+**	**#**	**&**	**Lib**	**Sol**
OBJECT TYPES FOR BASIC PROJECT STRUCTURING	PROJECT						
	STRUC	X	X	X	X	X	X
	DSTRUC				X		
	FSTRUC	X					
	LSTRUC					X	
	TSTRUC		X				
	MSTRUC		X				
OBJECT TYPES REPRESENTING PLANT FUNCTIONS	FSTRUC/	X					
	FGRP/	X					
	TAG/	X					
	DEV_FUN/	X				X	
OBJECT TYPES REPRESENTING ARTICLES	CUNIT		X			X	
	SUBUNIT		X			X	
	RACK		X			X	
	DEV	X	X			X	
	DCS_DEV		X			X	
	BAR		X			X	
	CABLE	X	X			X	
	ACCESSORY	X	X	X			
OBJ. TYPE REPR. DOCUMENTATION	DSTRUC				x		
OBJECT TYPES REPRESENTING CAD DOCUMENTS	HOOKUP	X				X	
	DRAWING	X	X	X		X	
	CI_DIA	X	X			X	
	PI_DIA	X				X	
	LA_DIA		X			X	
OBJ. TYPE REPR. TEXT DOCUMENTS	TEXT	X	X	X		X	

Table 1-1. AdvaBuild Object Types (Continued)

CLASSIFICATION	**OBJECT-TYPE**	**Assignment to Structure-types**					
		=	**++/+**	**#**	**&**	**Lib**	**Sol**
LIBRARY OBJECTS	LSTRUC		X			X	
	CUNIT		X			X	
	SUBUNIT		X			X	
	RACK		X			X	
	BAR		X			X	
	DEV	X	X			X	
	DCS_DEV		X			X	
	TERMINAL					X	
	CHANNEL					X	
	CPOINT					X	
	CABLE	X	X			X	
	CORE					X	

Table 1-1. AdvaBuild Object Types (Continued)

CLASSIFICATION	OBJECT-TYPE	Assignment to Structure-types					
		=	++/+	#	&	Lib	Sol
OBJECT TYPES FOR MP200, AC410,AND AC450 DB-ELEMENTS	Node/Station DB Element Object			X			
	Procesor Module Element Objects			X			
	Submodule Element Objects			X			
	Defining/Controlling Elements			X			
	Local General I/O Board Element Objects			X			
	Local Temperature I/O Board Element Objects			X			
	Pulse Counting I/O Board Element Objects			X			
	Distributed I/O Board Element Objects			X			
	Redundant I/O Board Element Objects			X			
	S800 I/O Module Element Objects			X			
	Communication Element Objects			X			
	Event Handling Objects			X			
	Application Oriented Objects			X			
	Function Unit Group 1 Objects			X			
	Function Unit Group 2 Objects			X			
	Function Unit Group 3 Objects			X			
	Signal Element Objects	X		X			
	BU Specific Objects			X			
OBJECT TYPES FOR AC110 DB-ELEMENTS	Processor Module Element Objects			X			
	Module Element Objects			X			
	Submodule Element Objects			X			
	Process I/O Board Element Objects			X			
	Signal Element Objects	X		X			
	Communication Element Objects			X			
	Event Handling Objects			X			
	Miscellaneous DB Element Objects			X			

Table 1-1. AdvaBuild Object Types (Continued)

CLASSIFICATION	**OBJECT-TYPE**	**Assignment to Structure-types**					
		=	**++/+**	**#**	**&**	**Lib**	**Sol**
OBJECT TYPES FOR AC70 DB-ELEMENTS	Processor Module Element Objects			X			
	Process I/O Board Element Objects			X			
	Signal Element Objects	X		X			
	Communication Element Objects			X			
	Event Handling Objects			X			
	Miscellaneous DB Element Objects			X			
OBJECT TYPES REPRESENTING NETWORKS, NETS AND BUSES	PNET			X			
	CNET			X			
	LCNET			X			
	TCP_IP			X			
	EXT_NET			X			
	GCOM			X			
	MB300			X			
	MB200			X			
	MVI_NET			X			
	RCOM			X			
	MB90			X			
	AF100			X			
	AF100IOS			X			
	AF100SC			X			
	MFB			X			
	EXCOM			X			

Table 1-1. AdvaBuild Object Types (Continued)

CLASSIFICATION	**OBJECT-TYPE**	**Assignment to Structure-types**					
		=	**++/+**	**#**	**&**	**Lib**	**Sol**
OBJECT TYPES REPRESENTING NODES WITHIN MASTER/ADVAN NETWORK	MODGATE			X			
	MG230			X			
	MV810			X			
	MV830			X			
	MV850			X			
	SVX00			X			
	MBC200			X			
	MP200			X			
	MP51			X			
	MP40			X			
	AC55C			X			
	AC110C			X			
	AC70C			X			
	AC410C			X			
	AC450C			X			
	AS100ES			X			
	AS100MS			X			
	AS510OS			X			
	AS515OS			X			
	AS520OS			X			
	AS515IMS			X			
	AS520IMS			X			
	AS800IMS			X			
	AS515ES			X			
	AS520ES			X			
	EXT_COMPUTER			X			
OBJECT TYPES FOR SOLUTIONS	SOLUTIONPOOL						X
	SOLUTION						X
SPECIAL PURPOSE	FREE	X	X	X			

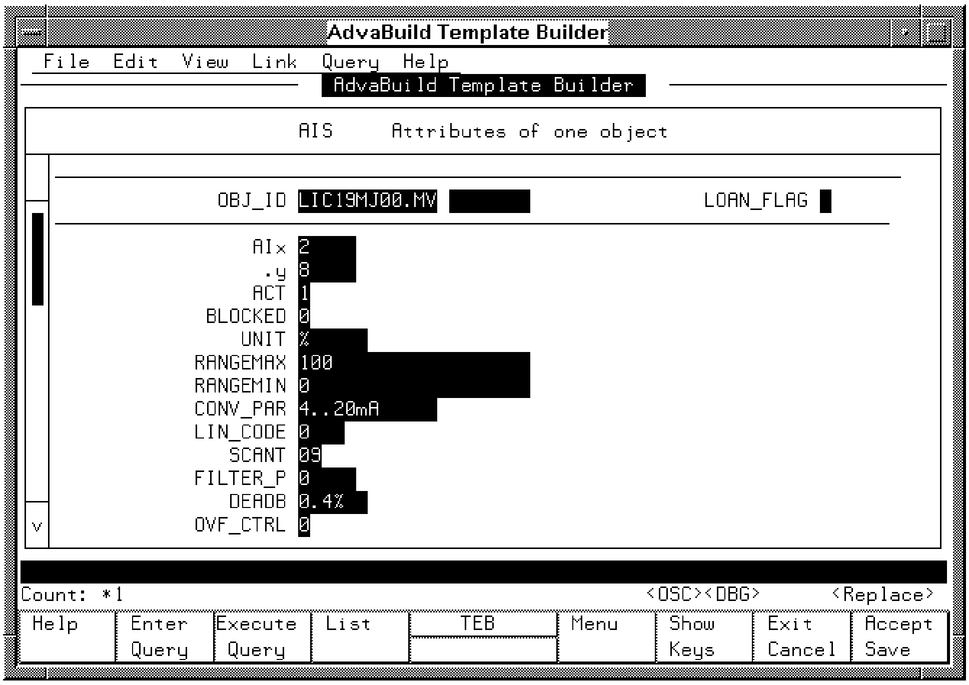
Table 1-2. AdvaBuild Link Types

LINK TYPE	CONNECTS	WITH
HAS_SIGNAL_ELEMENT	Channels of I/O Board Objects	Signal Element Objects
HAS_CONNECTION_TO	Signal Element Objects	PCPGM
IS_DOCUMENTATION_OF	DSTRUC	Objects in the function structure, location structure and control system structure
IS_IMPLEMENTED_BY	Board Objects	DCS_DEV
HAS_LOCATION	CUNIT	SUBUNIT RACK BAR DEV DCS_DEV
	SUBUNIT	RACK BAR DEV DCS_DEV
	RACK MSTRUC	BAR DEV DCS_DEV
	(DCS_)DEV	SUBMODULE
REFERENCES_SOLUTION	All object types, except SOLUTION and library objects	SOLUTION

1.8.1.2 Template Builder

The Template Builder is used to define object attributes via fill-in-the-blank templates. When you open the Template Builder view of an object via the Structure Builder, an interactive template with configurable fields for defining object attributes is displayed. There are two template formats. The default format displays one object of a selected type (single-object form), Figure 1-7 (top view). The multiple-object form displays multiple database objects of a selected object type for a selected subtree ,(for example all analog input signals) Figure 1-7 (bottom view). This facilitates bulk data entry, such as when creating a large number of objects with similar attributes. You can also insert, copy, and delete objects of a given type on the multiple-object form.

Single Object Form


AdvaBuild Template Builder
File Edit View Link Query Help
AdvaBuild Template Builder
Help
Enter Query
Execute Query
List
TEB
Menu
Show Keys
Exit Cancel
Accept Save

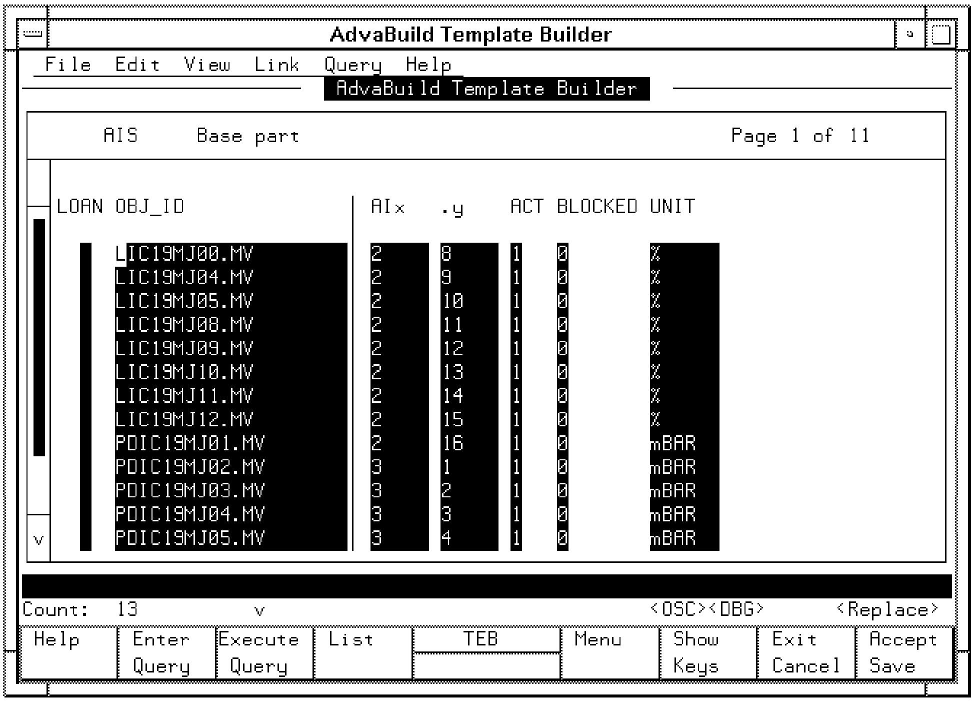
AIS Attributes of one object

OBJ_ID LIC19MJ00.MV LOAN_FLAG

AIx	2
.y	8
ACT	1
BLOCKED	0
UNIT	%
RANGEMAX	100
RANGEMIN	0
CONV_PAR	4..20mA
LIN_CODE	0
SCANT	09
FILTER_P	0
DEADB	0.4%
OVF_CTRL	0

Count: *1 <OSC><DBG> <Replace>

Multiple Object Form


AdvaBuild Template Builder
File Edit View Link Query Help
AdvaBuild Template Builder
Help
Enter Query
Execute Query
List
TEB
Menu
Show Keys
Exit Cancel
Accept Save

AIS Base part Page 1 of 11

LOAN	OBJ_ID	AIx	.y	ACT	BLOCKED	UNIT
	LIC19MJ00.MV	2	8	1	0	%
	LIC19MJ04.MV	2	9	1	0	%
	LIC19MJ05.MV	2	10	1	0	%
	LIC19MJ08.MV	2	11	1	0	%
	LIC19MJ09.MV	2	12	1	0	%
	LIC19MJ10.MV	2	13	1	0	%
	LIC19MJ11.MV	2	14	1	0	%
	LIC19MJ12.MV	2	15	1	0	%
	PDIC19MJ01.MV	2	16	1	0	mBAR
	PDIC19MJ02.MV	3	1	1	0	mBAR
	PDIC19MJ03.MV	3	2	1	0	mBAR
	PDIC19MJ04.MV	3	3	1	0	mBAR
	PDIC19MJ05.MV	3	4	1	0	mBAR

Count: 13 v <OSC><DBG> <Replace>

Figure 1-7. Template Forms

1.8.1.3 AdvaBuild Function Chart Builder

The Function Chart Builder is designed for off-line graphic configuration and documentation of AMPL (ABB MasterPiece Language) application programs. The Function Chart Builder can be used for:

- Advant Controller 100 Series application programs
- Advant Controller 400 Series application programs
- MasterPiece 200/1 application programs.

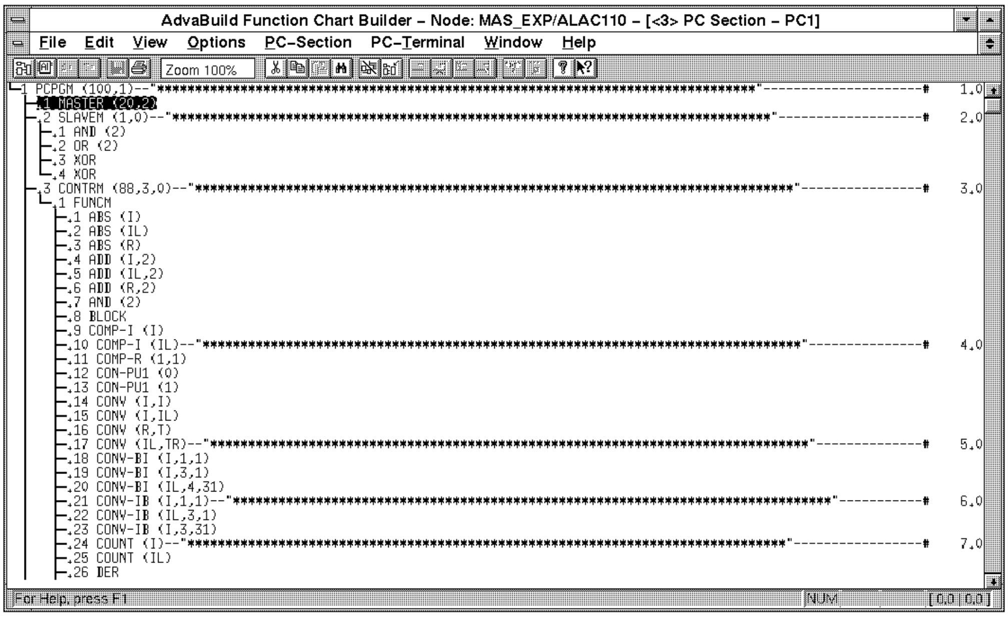
using the corresponding libraries.

Advant Controller 100 and 400 Series and MasterPiece 200/1 configuration, programming, documentation, source code generation and back translation are executed off-line, i. e. without being connected to a Controller.

The following functions are provided by AdvaBuild Function Chart Builder:

- Tree editor for control program structures
- Editing of function charts
- DB-Element editor
- PC (AMPL) source code editing and syntax checking in node mode.
- Direct access to the engineering database
- Symbolic addressing of signals
- Use of calculated symbols (parameters)
- Generation and back-translation of application programs and database source code. Can be used for transfer purposes between different process stations and engineering systems. The graphic representation after back-translation from the target station is the same as if the program was entered directly into the Function Chart Builder ensuring engineering consistency.
- Graphical documentation of application programs in function chart representation and in tree representation.
- Data base and cross reference documentation in list representation.
- Keyboard accelerators to speed up input and editing of data

Printer configuration, including postscript printing.


AdvaBuild Function Chart Builder – Node: MAS_EXP/ALAC110 – [<3> PC Section – PC1]
File
Edit
View
Options
PC–Section
PC–Terminal
Window
Help
Zoom 100%

```
└1 PCPGM (100,1)--"********************************************************************************"-------------------#    1.0
 ├.1 MASTER (20,2)
 ├.2 SLAVEM (1,0)--"********************************************************************************"------------------#    2.0
 │├.1 AND (2)
 │├.2 OR (2)
 │├.3 XOR
 │└.4 XOR
 ├.3 CONTRM (88,3,0)--"********************************************************************************"----------------#    3.0
 │└.1 FUNCM
 │  ├.1 ABS (I)
 │  ├.2 ABS (IL)
 │  ├.3 ABS (R)
 │  ├.4 ADD (I,2)
 │  ├.5 ADD (IL,2)
 │  ├.6 ADD (R,2)
 │  ├.7 AND (2)
 │  ├.8 BLOCK
 │  ├.9 COMP-I (I)
 │  ├.10 COMP-I (IL)--"********************************************************************************"---------------#    4.0
 │  ├.11 COMP-R (1,1)
 │  ├.12 CON-PU1 (0)
 │  ├.13 CON-PU1 (1)
 │  ├.14 CONV (I,I)
 │  ├.15 CONV (I,IL)
 │  ├.16 CONV (R,T)
 │  ├.17 CONV (IL,TR)--"********************************************************************************"--------------#    5.0
 │  ├.18 CONV-BI (I,1,1)
 │  ├.19 CONV-BI (I,3,1)
 │  ├.20 CONV-BI (IL,4,31)
 │  ├.21 CONV-IB (I,1,1)--"********************************************************************************"-----------#    6.0
 │  ├.22 CONV-IB (IL,3,1)
 │  ├.23 CONV-IB (I,3,31)
 │  ├.24 COUNT (I)--"********************************************************************************"-----------------#    7.0
 │  ├.25 COUNT (IL)
 │  ├.26 DER
For Help, press F1                                                    NUM          [0,0 | 0,0]
```

Figure 1-8. AdvaBuild Function Chart Builder, AMPL program in tree representation

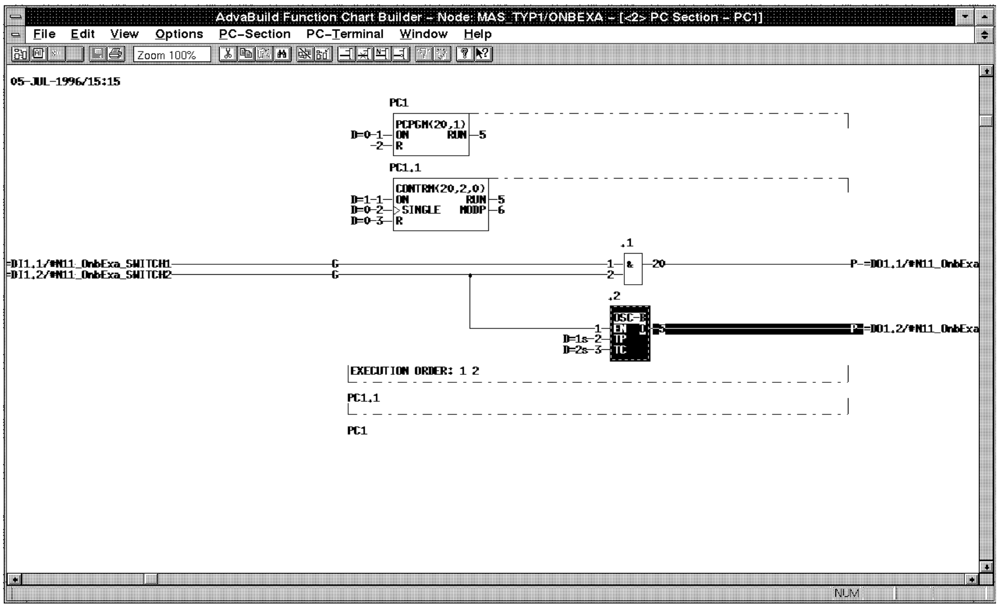


Figure 1-9. AdvaBuild Function Chart Builder showing an AMPL program in function-chart representation

1.8.1.4 SQL*ReportWriter Interface

SQL*ReportWriter interface provides the means to design and integrate custom reports for database documentation. These reports can be printed on an individual basis via **Print** and **Print Type** menu items in the Structure Builder, or they can be incorporated into and printed with document packages via the Plant Documentation Builder.

1.8.1.5 Plant Documentation Builder

The Plant Documentation Builder provides the means to build, manage, and print finished deliverable documentation packages for project engineering applications. Through the Plant Documentation Builder you collect individual documents that reside in the object hierarchy into documentation packages so they can be managed and printed as one unit. The Plant Documentation Builder generates a cover page and table of contents for each package automatically, and provides the means to label each package with revision information when you print it out. Project documentation handled by the Plant Documentation Builder includes:

- text generated via the Document Builder
- P&IDs, layout drawings, circuit diagrams generated via the Diagram Builder
- object, group, and process displays generated via the Display Builder
- function charts generated via the Function Chart Builder
- predefined and custom reports generated via the SQL*ReportWriter.

Documentation packages are handled as objects in the documentation (&) structure in the project database. You put together a documentation package for a specific part of the database by linking the appropriate objects with a Document object in the Documentation structure.

1.8.2 AdvaBuild On-line Builder

The AdvaBuild On-line Builder is an optional package of the Advant Station 500 Engineering Station. To support on-line functions, your workstation must be equipped with a Real Time Accelerator Board.

The AdvaBuild On-line Builder is similar to the On-line Builder of the Advant Station 100 Series Engineering Stations, including partial download of AMPL code. Some special functions for local mode operation, including the KERMIT and AMIS functions are, however, specific Advant Station 100 Series Engineering Station functions and not available in Advant Station 500 Series Engineering Station.

The On-line Builder may be used to load application programs via the MasterBus 300 into MasterPiece 200, MasterView 800, MasterGate 230 stations, AdvantController 400C, AdvantStation 500ES, AdvantStation 500OS, AdvantStation 500IMS and MasterBatch-Controller 200. Those programs may be tested and debugged with the On-line Builder, too.

Using the On-line Builder, you may:

- convert DB source code files
  for objects of MP200, PCPGM, MV810, MV830, MV850, MG230, AC410C, AC450C, AS515ES, AS520ES, AS510OS, AS515OS, AS520OS, AS515IMS, AS520IMS and MBC200 type.

- convert PC source code files
  for objects of MP200 and PCPGM type.
- use the On-line Builder Commands (like DIMDB, TRDBS, TRPCS).
- save and restore memory dumps of stations of the above DB object list.
- extract and convert PC source code from a big source code file.

In Advant Station 500 Series Engineering Stations the AdvaBuild On-line Builder should be used for small and fast modifications. At bulk data handling the AdvaBuild Template Builder or the Function Chart Builder are preferred. It is possible to work simultaneously with the Function Chart Builder and the On-line Builder on the same node.

The Advant Station 500 Engineering Stations, working towards MasterBus 300/300E, also give the possibility to run manual or scheduled backup of Master Piece 200/1, MasterView 800/1 Series, or MasterGate 230/1 nodes. The configuration is effected by script files.

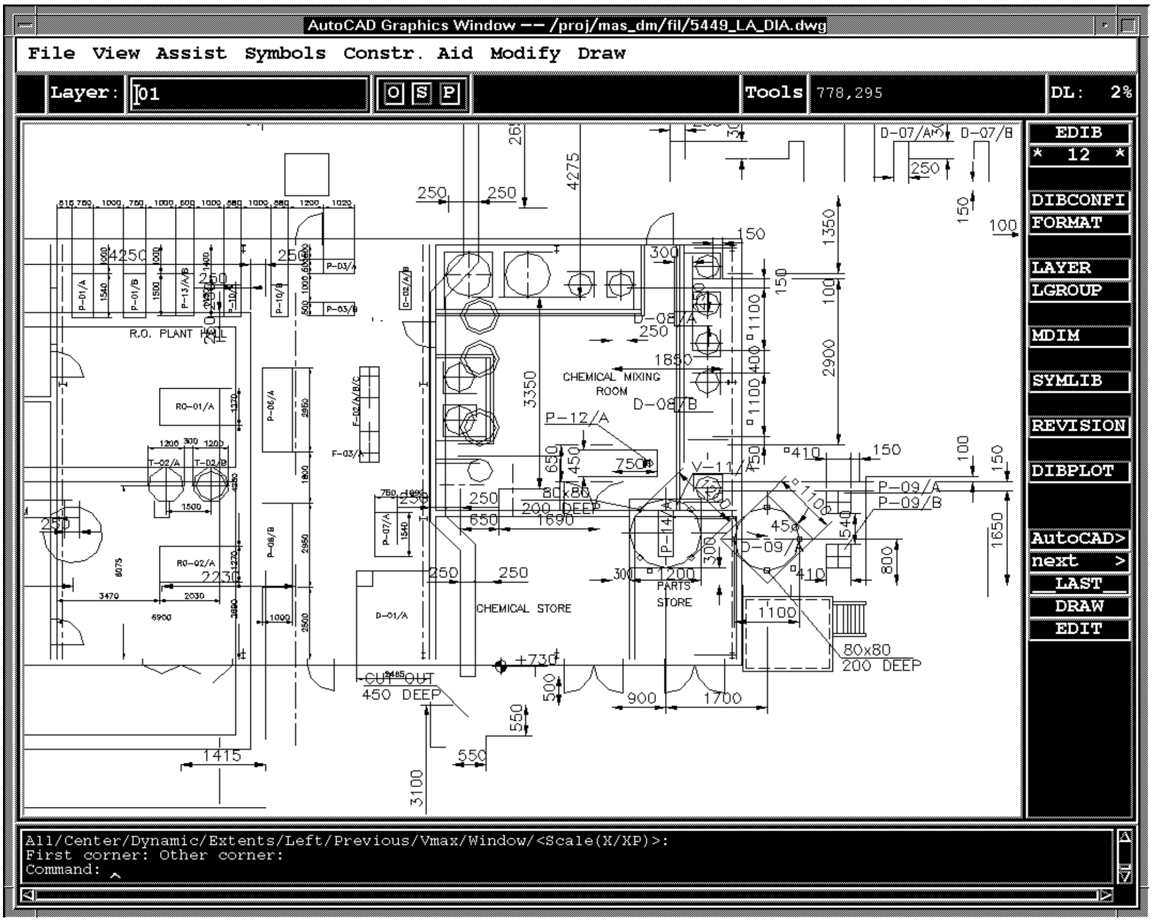
1.8.3 Import/Export Utility

The Import/Export utility provides the means to import data from an ASCII file into the AdvaBuild database, and to export data from the AdvaBuild database to an ASCII file.

1.8.4 Optional Builders

1.8.4.1 AdvaBuild Diagram Builder

The Diagram Builder uses AutoCAD version 12 to provide general-purpose CAD functionality. AutoCAD is a powerful drawing tool, supporting the creation of just about any engineering drawing or diagram, Figure 1-10. Import of graphics made in commonly used formats, like DXF and IGES is supported. Symbol libraries can be configured project specific.



```
All/Center/Dynamic/Extents/Left/Previous/Vmax/Window/<Scale(X/XP)>:
First corner: Other corner:
Command:
```

Figure 1-10. AdvaBuild Diagram Builder, Plant Layout

1.8.4.2 AdvaBuild Electrical Diagram Builder

The AdvaBuild Electrical Diagram Builder is based on the AdvaBuild Diagram Builder. It provides additional functionality, including drawing support functions from screen or digitizer menus to create and edit electrical circuit diagrams and layout drawings. The functionality includes:

- Multiple connection line insert
- Automatic break up or closing of single and multiple connection lines at symbol insertion/deletion
- Stretching of connection lines when moving symbols
- Terminal handling (semiautomatic drawing support)
- Selection of symbols from a graphical menu
- Handling of boundary frames.

Different symbol libraries are provided for different disciplines, for example:

- Electrical equipment, such as relays, terminals, breakers
- Instrumentation and control components, such as sensors, actuators, transformers, I/O channels
- Plant and process components, such as tags, valves, vessels
- Control system components, such as cabinets, racks, boards.

The symbol building utility helps to build new or modify existing symbols in the correct manner. It also supports the integration of a new symbol into the pop-up menu on the screen. Symbols are stored in an application specific symbol library. An example of a drawing produced with the Electrical Diagram Builder is shown in Figure 1-11.

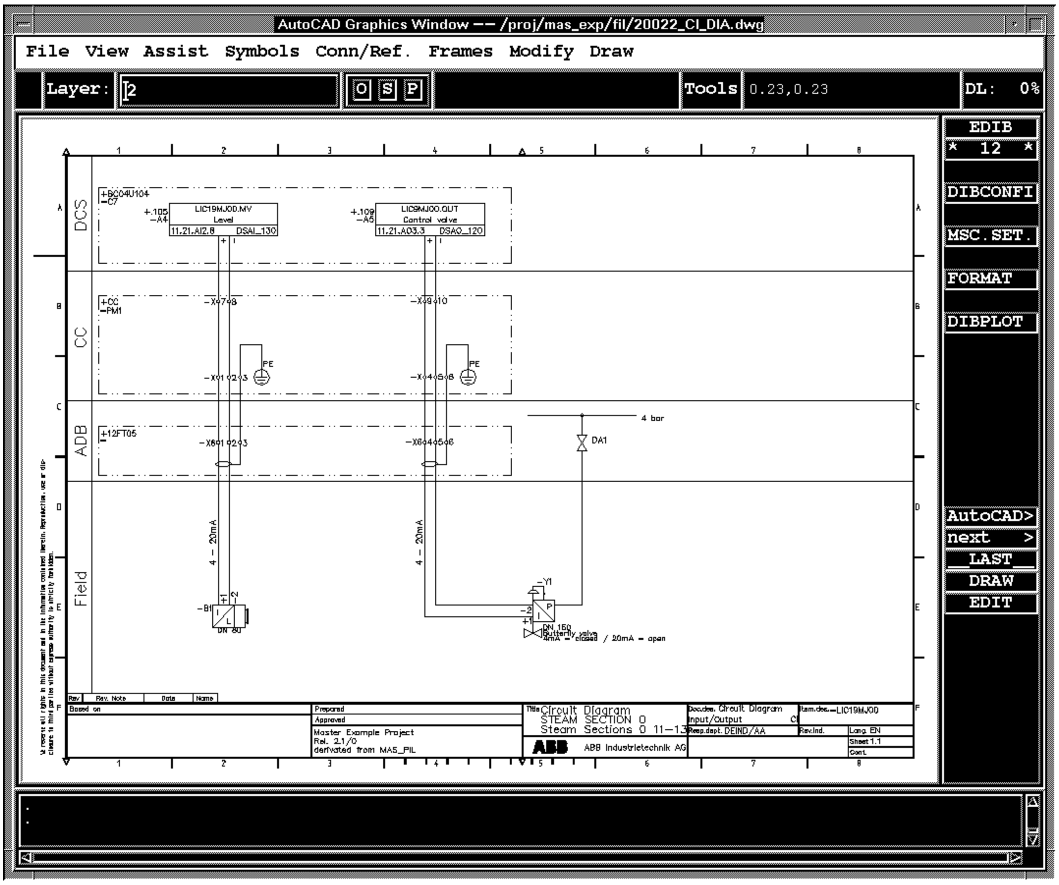


Figure 1-11. AdvaBuild Electrical Diagram Builder, Circuit Diagram

1.8.4.3 AdvaBuild Display Builder

AdvaBuild Display Builder is a menu-driven tool for the creation of graphical displays for process supervision and control via AdvaCommand User Interface. Display Builder´s true-image presentation of the ongoing work, its powerful features and its intuitive style of operation makes display creation efficient and productive.

The Display Builder is based on the principle WYSIWYG (What You See Is What You Get). It uses menu bars with pull down menus, palettes, and dialog boxes, sometimes with option menus.

All displays are built with static graphical information and dynamic graphical objects called display elements. The objects in the process and the system are represented by display elements on the display. New display elements can be created and existing elements can be edited.

Each display element is a view of one or more process or system objects (e.g. an analog input signal, a pump, a valve, and so on.). It provides information about the objects, and defines the interaction between them and the user. A display element can be as simple as a numerical value or as complex as a complete object display for a process object.

Display elements contain static and dynamic parts for presentation, an interface part and a data subscription part for data exchange with the objects.

The static background is created by selecting drawing tools and colors from palettes called up through pull-down menus. The engineering efficiency is further boosted via powerful copy functions and support for clip-books or symbol libraries.

The appearance of dynamic points is chosen by selecting display elements from an iconized menu and filling in forms that tie the points to the desired objects in the process database. An example of a display build using the Display Builder is shown in Figure 1-12.

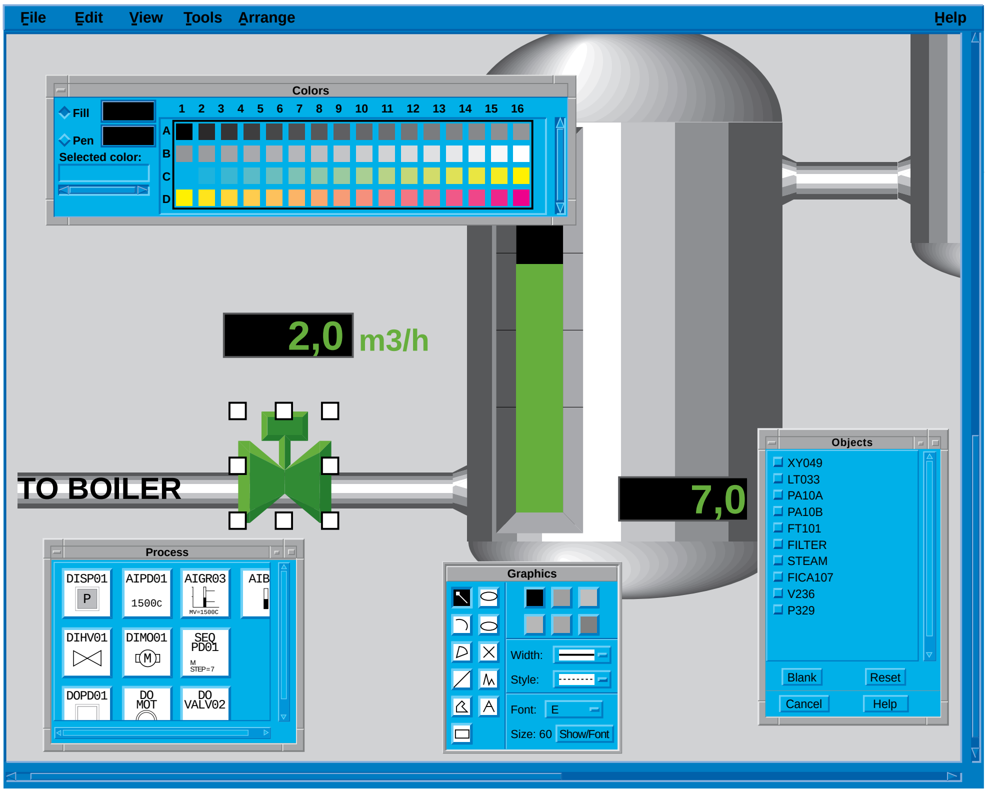


Figure 1-12. AdvaBuild Display Builder with Color and Tool Selection Menus Open

Displays can be configured as follows:

- Choose size and layout of the display.

  This is done by choosing from a list of ready-made templates. The templates are specially designed for process displays, group displays and fixed displays.

- State how the process objects shall be displayed by means of Display Elements.

  The display elements are displayed as icons in ready-made or user-designed palettes.

- Connect the process objects to the Display Elements.

  There are two ways of doing this. The easiest and quickest way is to import ASCII files which contain the object names. You can use one object list for the whole system or you can use one object list for each display. You select the objects by clicking with the select mouse button on the object name and then click on a display element in a palette. Built-in markers near the names in the object list indicate that the objects have been picked out. This helps you to check that you have not accidentally forgotten to include an object on a display. The second way to make the connection is to call up an attribute box for the display element and type the object name in the box.

- Position the Display Element in the display

  You click in the work area directly after choosing the display element in the palette.

- Set the Display Element Attributes, if default settings are not suitable.

  All display elements have default settings for update frequency, process dialog, color, and size, but they can easily be changed. The display elements in the display element library are not affected, only the selected display element in the display.

- Create the static information, such as lines and graphics.

  The Graphic Palette provides graphical objects as well as fill pattern and texture. You choose color from the Color Palette which contains 192 colors. The colors, line attributes, fill pattern and font, which will be used throughout for a special purpose, for example symbolizing Water, Oil, Main Flow and Secondary Flow, can be predefined and selected from a named list. This simplifies the attribute settings and provides displays of uniform appearance. Professionally made composite graphics are also included in a separate list to help the display designer.

- Set the Display Attributes.

  The attributes include the background color, display type, linked displays, and dynamic function keys.

- Name, save and install the display.

Existing displays can be edited or deleted. New displays can be created by copying existing displays (in whole or in part), editing them, and storing them under new identities.

The Advant Station 500 Series Engineering Stations also include functions to download displays to an Operator Station over the Plant Network. Similarly, displays can be uploaded to the Engineering Station.

1.8.4.4 AdvaBuild Document Builder

The Document Builder uses FrameMaker as basis to create free-styled documents. FrameMaker provides a full set of tools for documentation, including a full-feature word processor, an advanced graphic toolbox and many unique document handling utilities.

The following functions are provided by AdvaBuild Document Builder:

- WYSIWYG (What-You-See-Is-What-You-Get) interface
- Full-feature word processor
- Graphics creation, editing and import capabilities
- Capture facility for any X-Window screen, (also from other applications such as Structure Builder, Template Builder, and so on.) This makes it possible to include such screen dumps into the plant documentation from the Document Builder.
- Mathematical equation composition
- Special features for management of large documents
- Table handling
- Comparison of different document versions.

A number of ABB templates are included, for example: for memo, letter, fax, overhead, user manual. User-defined templates can easily be created.

By means of the included filter package, document and graphics files in other formats can be converted in order to be used in the Document Builder. Following formats are supported, Table 1-3.

Table 1-3. Import and Export Formats

Format	Import	Export
IBM Document Content Architecture Revisable Form - Text (DCA)	yes	-
Interleaf ASCII Format (IAF) 6.4 (TPS 4.1)	yes	yes
Microsoft Word 3.0, 4.0 and 5.0 (Word) for the Macintosh and Microsoft Word for Windows 1.0 and 2.0 (except Graphics)	yes	-
troff (pure, -me, -ms, and -man)	yes	-
WordPerfect 5.1	yes	yes
Computer Graphics Metafile (CGM)	yes	yes
AutoCAD Interchange Format (DXF)	yes	-
Hewlett-Packard Graphics Language (HPGL)	yes	-
Initial Graphics Exchange Specification (IGES) Class I or Class II formats	yes	-
CCITT Group 4	yes	yes
Encapsulated PostScript Interchange (EPSI)	yes	-
Macintosh PICT graphics	yes	-
Rich Text Format (RTF)	-	yes
CalComp 960	yes	-

1.9 User Interface for AdvaBuild Basic Software

AdvaBuild basic functions are supported by two basic user interfaces. The general user interface supports the Structure Builder and Function Chart Builder, and is based on HP Visual User Environment (HP VUE). The Function Chart Builder can be switched between MS-Windows and OSF-Motif look, by default ist is the MS-Windows look.

The FORMS user interface supports the AdvaBuild Template Builder, AdvaBuild Plant Documentation Builder, Link Handling, and the Administration function. This user interface is based on SQL*FORMS.

1.9.1 General User Interface

This section provides an overview of the HP Visual User Environment (HP VUE) methods incorporated into the Advant Station user interface. For further information, refer to the *HP Visual User Environment User's Guide.*

1.9.2 Using a Mouse

The mouse is a cursor control device which allows cursor movement in any direction.

To **click** a mouse button, press and release the button quickly.

To **double-click** a mouse button, press and release the button twice quickly.

The **pointer** on the screen indicates the current mouse location, and shows where on the screen mouse actions take effect. To move the pointer, move the mouse in the corresponding direction.

To **drag** an object on the screen:

1. Move the pointer over the object.
2. Press and hold down the mouse button.
3. Move the mouse until the object is at the desired new location.
4. Release the mouse button.

Recognizing Pointer and Cursor Shapes

The shapes of the pointer and cursor change according to location and function, as shown in Table 1-4. (The cursor indicates the focus of keyboard input.)

Table 1-4. Pointer and Cursor Shapes

Pointers (Mouse)	Cursors (Keyboard)
Working: ⌛	Inserting: I ^ ▮
Moving: ✥	Selecting: OK OK
Pointing: ↖ ×	
Resizing: ⇱ ⤒ ⇗ ⇤ ⇥ ⇙ ⤓ ⇲	
Help: ?	
Grabbing: +	

1.9.3 Working with Windows

A window is any screen element (dialog box, directory, message box, and so on) surrounded by a frame that is displayed as a result of user interaction with the system. Figure 1-13 shows an example of a window and identifies its components.

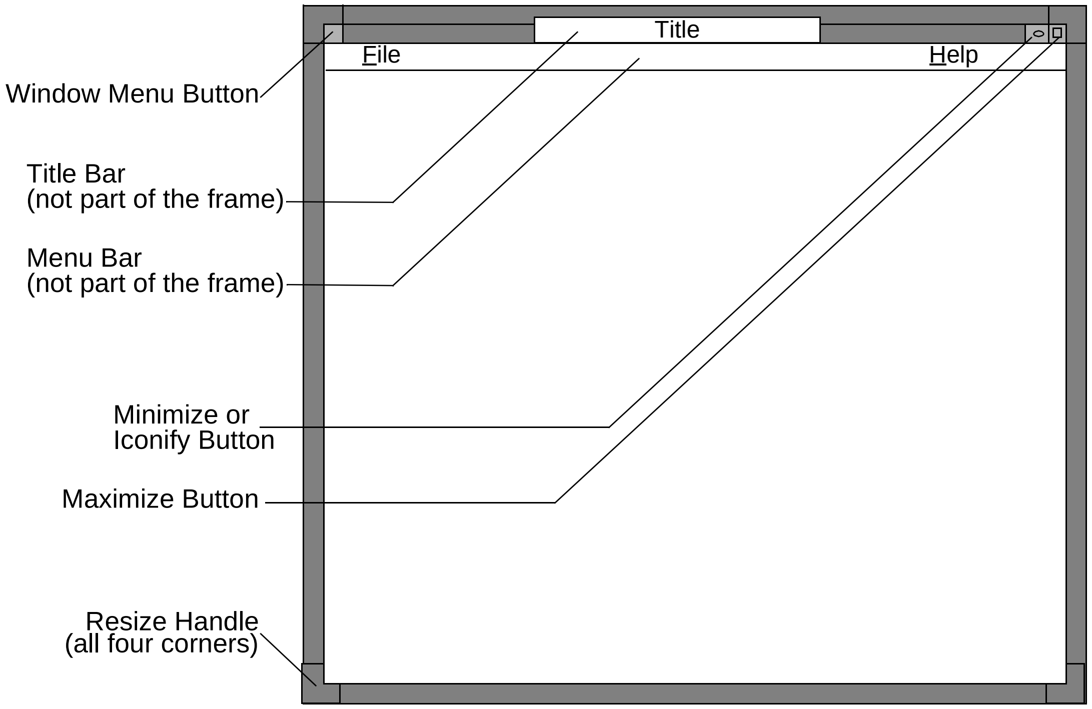


Figure 1-13. Window Frame

You can have multiple windows open, but only one can be active at a time. To make a window active, click anywhere on the window. The active window is indicated by its highlighted frame and title bar. To use window buttons:

- Click on the minimize (iconify) button to turn the window into an icon, or small graphic representation. This unclutters the workspace. Double-click on the icon to restore the window.
- Click on the maximize button to enlarge the window to its largest possible size.

CAUTION

DO NOT use the Close button within Window Menu at the top left side of windows to close the window. This may abort the currently running program and therefore leave the database in an inconsistent state! If you want to iconize the window, use the Motif *Minimize* function.

To move a window:

1. Move the pointer over the title bar.
2. Press and hold down mouse button one or two.
3. Drag the window to a new location.
4. Release the mouse button.

To resize a window:

1. Move the pointer over the window frame or a resize handle. Resizing from the window frame changes either the width or the height of the window depending on the location of the pointer, and resizing from a resize handle changes both the width and the height.
2. Press and hold down mouse button one or two.
3. Drag window frame to new size.
4. Release the mouse button.

To shuffle layered windows, click anywhere on the visible portion of a partially concealed window to move it to the top.

Using the Window Menu

Each application window has a window menu in the upper left corner of the window frame, which enables you to control the size, shape, and position of the window. To display the window menu, click on the window menu button.

The window menu contains the following items:

Restore	Turns an icon back into a window.
Move	Interactively moves the window.
Size	Interactively changes the size of the window.
Minimize	Turns the window into an icon.
Maximize	Enlarges the window to its greatest allowable size.

Lower	Puts the window on the bottom of the stack of overlapping windows on the workspace.
Occupy...	Opens a dialog box to select the workspaces in which the window should appear.
Occupy all	Displays the window in all workspaces.
Remove from WS	Removes the window from the current workspace.
Close	Closes the window, removing it from all workspaces.

Keyboard Focus

More than one window can be displayed at the same time; however, only one window is active at a time. Keyboard focus determines which area of the screen or which element in a window gets input from the keyboard. The active window is indicated by its highlighted frame and title bar. The location cursor indicates which element on the screen has keyboard focus. The location cursor is usually a box around a default selection element or a text insertion cursor within a text field. When applicable, pressing <**Return**> executes the default operation.

Using a Scroll Bar

A scroll bar, Figure 1-14, is used to traverse through text and graphics in a viewing area (such as a document or a menu). Scroll bars can be either horizontal (to scroll left and right) or vertical (to scroll up and down).

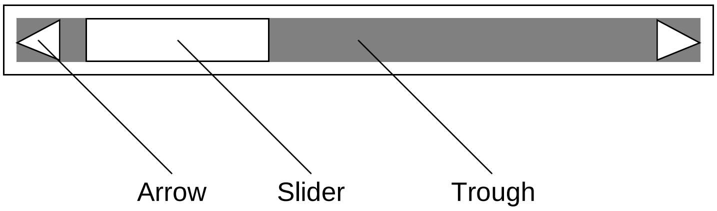


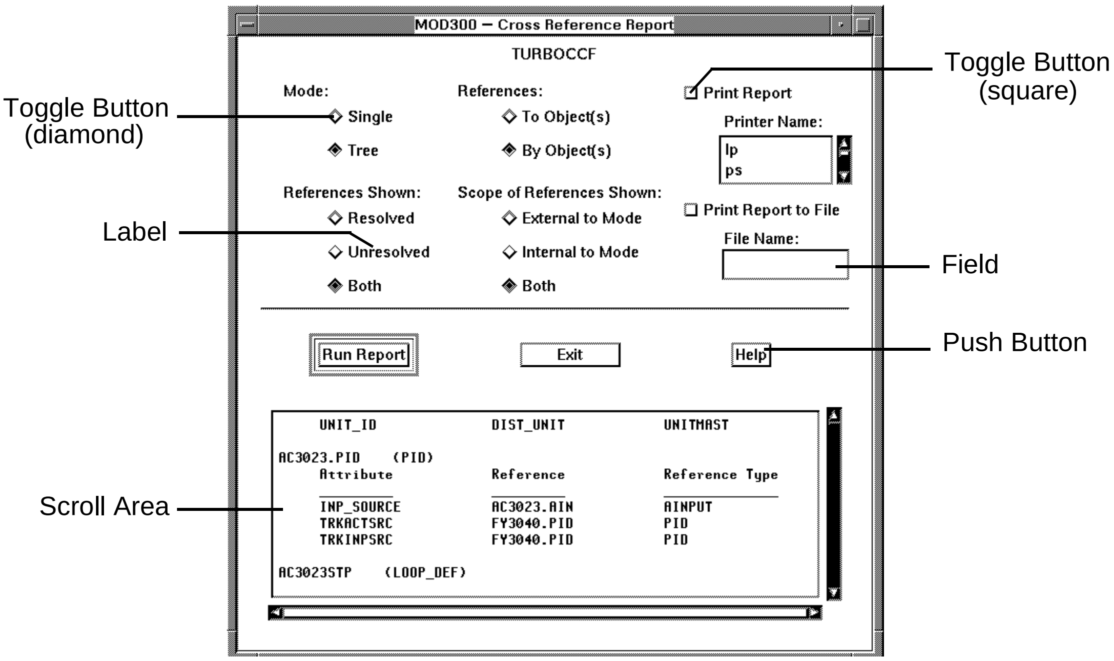
Figure 1-14. Scroll Bar

To scroll through a window:

- Click on one of the arrows to scroll one line at a time in the corresponding direction.
- Click in the trough on either side of the slider to scroll one page at a time in the corresponding direction.
- Drag the slider to scroll the list incrementally. The ratio of the slider size to the trough size corresponds to the ratio of the number of items displayed in the viewing area to the total number of items in the list.

1.9.3.1 Using a Dialog Box

A dialog box, Figure 1-15, is a window used for the entry of user input. It can contain any number of fields, labels, buttons, and/or scroll areas.



Mode:
◇ Single
◆ Tree

References:
◇ To Object(s)
◆ By Object(s)

☑ Print Report

Printer Name:
lp
ps

References Shown:
◇ Resolved
◇ Unresolved
◆ Both

Scope of References Shown:
◇ External to Mode
◇ Internal to Mode
◆ Both

☐ Print Report to File

File Name:

```
     UNIT_ID              DIST_UNIT              UNITMAST

AC3023.PID    (PID)
     Attribute            Reference              Reference Type
     _________            _________              ______________
     INP_SOURCE           AC3023.AIN             AINPUT
     TRKACTSRC            FY3040.PID             PID
     TRKINPSRC            FY3040.PID             PID

AC3023STP    (LOOP_DEF)
```

Figure 1-15. Dialog Box

Field

A field is an area that the user can select and enter data into. You select a field by clicking on it. When a field is selected, it becomes highlighted and a cursor is displayed in the field.

Label

A label is text in a window or on a button that provides information but cannot be edited.

Button

All selectable buttons are depicted as raised above the plane of the screen. You select a button by clicking on it. When a button is selected, it appears lower than the plane of the screen.
All nonselectable areas of the screen appear flat (in the plane of the screen).

A push button initiates a function, such as accepting user input, opening another window, starting a program, and so on. The function initiated is usually indicated by the label on the button.

A toggle button indicates a state. There are two indicator shapes: diamond and square. The diamond-shaped indicator signifies that only one toggle button in a group can be selected at a time. The square-shaped indicator signifies that any number of toggle buttons in a group can be selected at a time.

1.9.3.2 Menus

To display a menu, click on the menu name in the menu bar. This displays the list of menu items for the selected menu, Figure 1-16. Then click on the desired menu item. All menus are pop-up (not spring-loaded), so the menu remains open until an action is taken or the menu is explicitly closed. You do not have to hold the mouse button down after clicking to keep the menu displayed. However, you can click and hold on the menu and then drag the cursor down to the desired menu item if you want. When the desired item is highlighted, release the mouse button.

As an alternative, you can display a menu by holding down <**Extend char**> or <**Alt**> and then pressing the key corresponding to the character underlined in the menu name (for example, <**Extend char-e**> for the **Edit** menu). This shows the menu items. Then you can choose a menu item by pressing the key corresponding to the underlined character in the menu item name (for example, <**C**>for **Copy**).

Accelerator keys are implemented for some menu items. An accelerator key is a key combination that invokes the action of a menu item when the accelerator is pressed. The accelerator key is displayed to the right of the menu item and is used in combination with <**Extend char**> or <**Alt**>. The menu does not have to be displayed to use an accelerator.

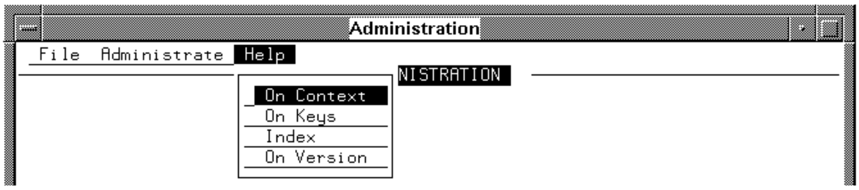
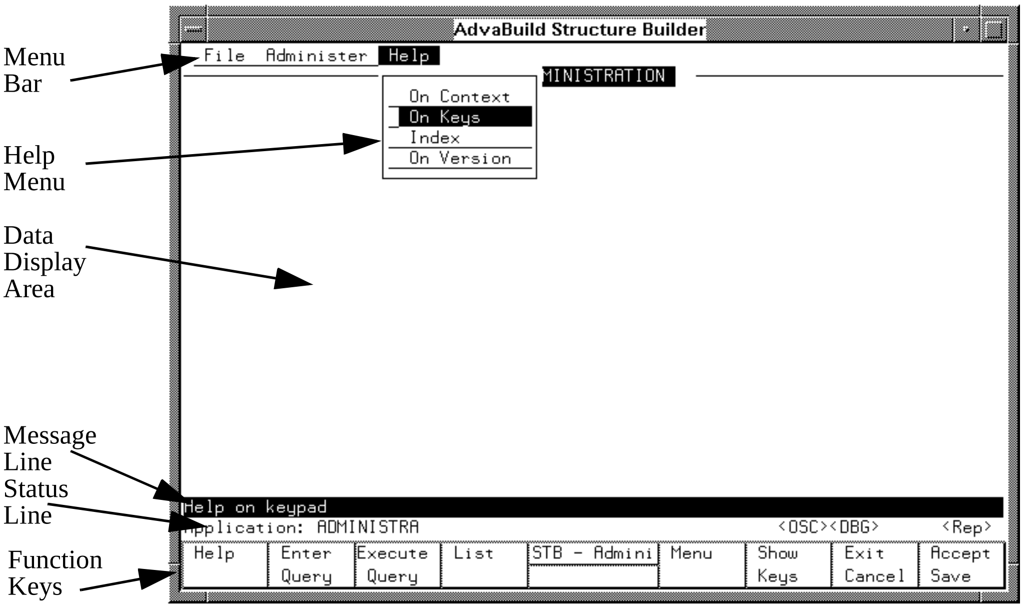


Figure 1-16. Menu for AdvaBuild Administration

1.9.4 FORMS User Interface

The FORMS user interface is used in the Project Administration function, Template Builder, Link Handler, Plant Documentation Builder, and the Object Information window. The basic layout for this user interface is shown in Figure 1-17.



Help on keypad

Application: ADMINISTRA <OSC><DBG> <Rep>

Figure 1-17. FORMS General Layout

In this user interface, mouse selection is limited to the function keys along the bottom line of the window. All other operations are via keyboard functions and menu items. A listing of all keyboard functions is provided via the **Show Keys** function (<**f6**>).

1.9.4.1 Function Keys

You can select any function on the bottom line either by pressing the associated function key (<**f1**> through <**f8**>), or by moving the cursor over the function key target on the screen and clicking the left mouse button. The purpose and operation of these functions are described below.

Help (<f1>)

Help displays help text for the currently selected item (indicated by cursor position).

Enter Query (<f2>)

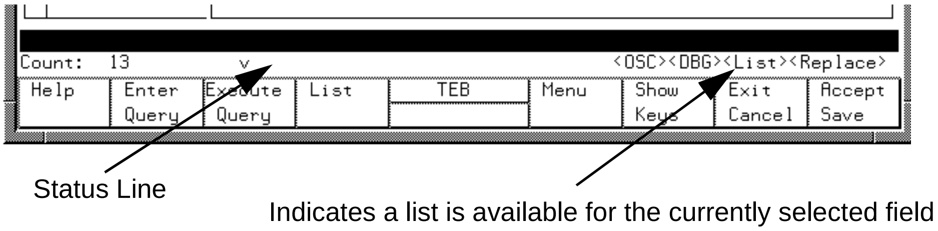
Enter Query sets the current window for entering a query. Refer to Section 3.4.11, Queries for further information regarding query functions.

Execute Query (<f3>)

Execute Query executes the query defined via the **Enter Query** function.

List (<f4>)

List shows a pre-defined list of values for the current field. This function is not available for all fields. When a list is available for a selected field, it is indicated in the status line, Figure 1-18.



Status Line

Indicates a list is available for the currently selected field

Figure 1-18. List Function

Use <**Tab**> to navigate through the list. When the cursor is located on the desired value, select the **Accept/Save** function key. This closes the List of Values window and places the selected value in the appropriate field.

You can close the List of Values window without selecting a value by selecting the **Exit/Cancel** function key.

Menu (<f5>)

Menu moves the cursor up into the menu bar. For further information regarding operation of menus, refer to Section 1.9.4.6, Menus.

Show Keys (<f6>)

Show Keys provides a listing of all keyboard functions. To exit this display and return to the original display, press <**Return>**

Exit/Cancel (<f7>)

Exit/Cancel cancels the selected action or closes the window. Using this key when data has been entered opens a window where the changes can be committed or discarded.

Accept/Save (<f8>)

Accept/Save commits your current entries into the database, and/or starts the currently selected function. When the function is lengthy, a warning message is generated. After the function has finished, the message must be acknowledged by pressing <**Return**>.

1.9.4.2 Special Key Functions

First Record

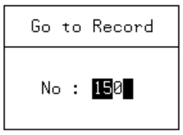
This function moves the cursor from any record position to the first record without leaving the same column. To invoke First Record: press and release <**ESC**>, press and release <**ESC**> again, and then press **0** (the number zero). You can use the **HOME** key in place of **0**.

Last Record

This function moves the cursor from any record position to the last record without leaving the same column. To invoke Last Record: press and release <**ESC**>, press and release <**ESC**> again, and then press **1** (the number one).

Go To Record

This function moves the cursor from any record position to a specified record without leaving the same column. To invoke Go To Record: press and release <**ESC**>, press and release <**ESC**> again, and then press **2** (the number two). This displays the Go To Record dialog box, Figure 1-19**.** Use this box to specify the number of the record that you want to move the cursor to. If the number you enter is greater then the number of records displayed in the form the cursor moves to the last record.



Go to Record

No : 150

Figure 1-19. Dialog box "Go To Record"

Toggle Form

This function toggles between the multiple object form and the single object form. To toggle between the forms: press and release <**ESC**>, press and release <**ESC**> again, and then press **3**.

Next Part/Previous Part

These functions move the cursor to the first attribute in either the next part or the previous part. To move to the next part press <**Next**>. To move to the previous part press <**prev**>.

Scroll Down/Scroll Up

These functions scroll the form's list of records up (cursor goes down) or down (cursor goes up) by approximately 80 percent of the form's display. To scroll down press and hold the Shift key, and press the down arrow. To scroll up press and hold the Shift key, and press the up arrow.

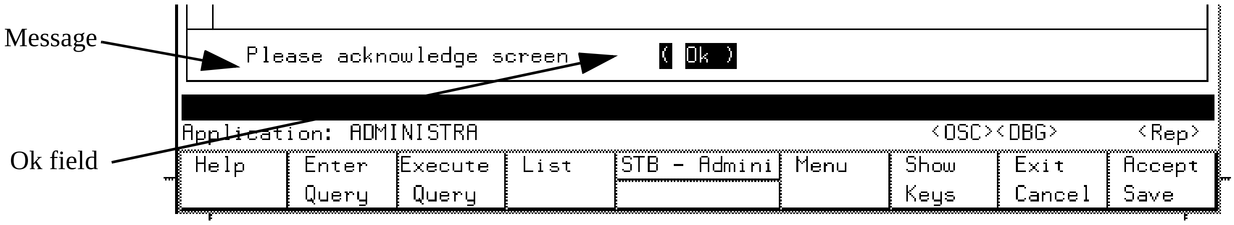
1.9.4.3 Status Line

The status line above the function keys displays the application name and shows status symbols that indicate the active modes of the window. The following mode symbols are defined:

<List>	Indicates that there is a list of values for the current field.
<Ins> or <Rep>	Indicates the editing mode: **Ins**ert or **Rep**lace.
<OSC>	Indicates that authority to issue operating system commands is granted.
<DBG>	Indicates that access to the debug mode is granted.
ENTER QUERY	Indicates that the query mode has been activated.

1.9.4.4 Message Line

The message line above the status line displays error messages and other miscellaneous information for a chosen function, Figure 1-20. Some messages have to be acknowledged by pressing <**Return**>. The need to acknowledge a message is indicated by an `OK` message with a prompt to acknowledge, Figure 1-20. Do not press <**Return**> in rapid succession; otherwise you may miss a message.



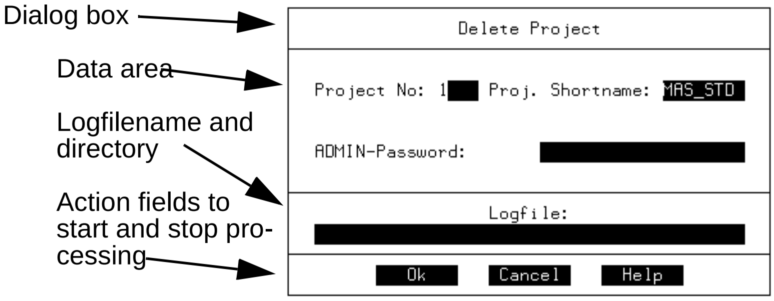
```
Please acknowledge screen     ( Ok )
Application: ADMINISTRA                    <OSC><DBG>      <Rep>
```

Figure 1-20. Message Line

If a message contains information about directory names or environment variables, it can be selected and copied with the mouse and then put into a command string in a terminal window.

1.9.4.5 Data Area and Dialog Boxes

The area above the message line is used to display the window-specific data or a dialog box where data can be entered, Figure 1-21.



Delete Project

Project No: 1 Proj. Shortname: MAS_STD

ADMIN-Password:

Logfile:

Dialog box
Data area
Logfilename and directory
Action fields to start and stop processing
Ok
Cancel
Help

Figure 1-21. Example of a Dialog Box

Most dialog and selection boxes have the following three buttons:

- **OK** is used to activate the function represented by the dialog box. After entering the necessary input in the box, select **OK** and then select **Accept/Save**.
- **Cancel** is used to cancel all entries in the current box before program execution.Select **Cancel** and then select **Accept/Save**.
- **Help** is used to select on-line help. Navigate to **Help** and select **Accept/Save**.

You navigate between fields in a dialog box via <**Return**> or <**Tab**> (<**Shift-Return**> or <**Shift-Tab**> moves the cursor backward). Cursor movement within a field is via the arrow keys. Generally, if a field requires an entry, the system will not allow you to advance the cursor to the next field until the entry has been made. Some functions are password-protected. In these cases, you are required to enter the ADMIN password. The password is not echoed (shown on the screen) when you enter it. After you make the final mandatory entry, the cursor is located on the **OK** button. Press <**f8**> (**Accept/Save**) with the cursor on **OK** to commit your entries.

1.9.4.6 Menus

The top line of the window contains the menu bar. To access the menu bar press <**CTRL-g**> or the **menu** function key (<**f5**>). To select a menu, use the cursor keys to position the cursor over the menu and then press <**Return**>. This opens the menu and shows the menu items. To choose a menu item, use the cursor keys to position the cursor over the menu item and then press <**Return**>. To exit the menu bar, press <**CTRL-b**> or the **Exit/Cancel** function key (<**f7**>).

Chapter 2 Installation

2.1 Site Planning/Environment

This section is not applicable.

2.2 Setup

AdvaBuild software and applicable libraries are installed at the factory prior to shipment. Instructions for installing AdvaBuild software and applicable libraries are provided in the Advant Station Engineering Station Release Notes.

2.3 Power Up Procedure

To power up the Engineering Station turn on the power to the Engineering Station and peripherals in the following order:

1. Uninterruptible Power Supply (UPS)
2. Monitor
3. External disk drives and DAT drives
4. Printers
5. System Unit.

2.4 Start-up Procedure

The procedures described in this section are intended for system administration-level users.

Use this procedure to log into a project when you power up the Advant Station. If the Advant Station is already powered up, you can use the procedure described in Section 3.1, Application Start-up.

To start up the AdvaBuild application, after powering up the Engineering Station:

6. Log in to HP VUE as the master user.
7. Enter **`y`** to start the Real-time Accelerator, or **`n`** if you do not want to start the Real-time Accelerator at start-up question in the hpterm.
8. In terminal window enter: **`db start`**

   This command starts the Oracle database.
9. Enter **`stb &`** to start the Structure Builder.
10. Login to a project as described in Section 3.1.2, Logging In to a Specific Project or the Administration Function.

2.5 Shut-down Procedure

The procedures described in this section are intended for system administration-level users.

Use this procedure to power down after finishing your AdvaBuild session. To simply exit a project without powering down, just choose **Exit** from the **File** menu in the Structure Builder.

To prepare for powering down when you exit a project:

1. Choose **Exit** from the **File** menu in the Structure Builder.
2. In the terminal window enter: **`db shuti`**

   This shuts down the Oracle database.
3. Close the terminal window and exit the VUE session.

   The HP VUE Login window is opened when you exit the VUE session.

2.5.1 Power Down the Engineering Station

To power down the Engineering Station:

1. In the HP VUE Login window, click **Options**.
2. Select **No Windows**.

   This turns off the graphical interface and displays the login prompt in the lower left corner.
3. Enter: **`poweroff`** and then press <**Return**>.
4. At the `password` prompt, enter the password (default is `po`) and then press <**Return**>.

   Some messages will be displayed, ending with the following:

   ```
   Halted - you may now cycle power
   ```
5. When the shutdown has finished, turn power off at the Engineering Station and peripherals in the following order:
   a. First turn off the System Unit, then printers.
   b. Then external mass memory system and DAT drives, then monitor, and finally the UPS.

WARNING

Do not assume that all power to the computer is off just because the power on/standby switch has been turned off. To completely remove power from your computer, unplug the power cable from the power outlet and disconnect that cable from the power supply.

Chapter 3 Configuration/Application Building

3.1 Application Start-up

To start up an application for AdvaBuild basic functions, you must first log in to HP-UX, and then log in to either a specific project, or the Administration function.

3.1.1 Log In To HP-UX

Seven pre-defined HP-UX user name/password sets are provided with the Advant Station; one for global access including administration functions, and six for configuring Master databases. You can change these passwords or define additional user names and passwords through HP-UX, Table 3-1.

Table 3-1. Application Users

User	Password	Usage
cape	cape	Administration, internal ORACLE user
master	master	Master application basic user
master1	master1	Master project user1
master2	master2	Master project user2
master3	master3	Master project user3
master4	master4	Master project user4
master5	master5	Master project user5

When you power up the Advant Station, the HP-UX Log In dialog box is displayed, Figure 3-1.

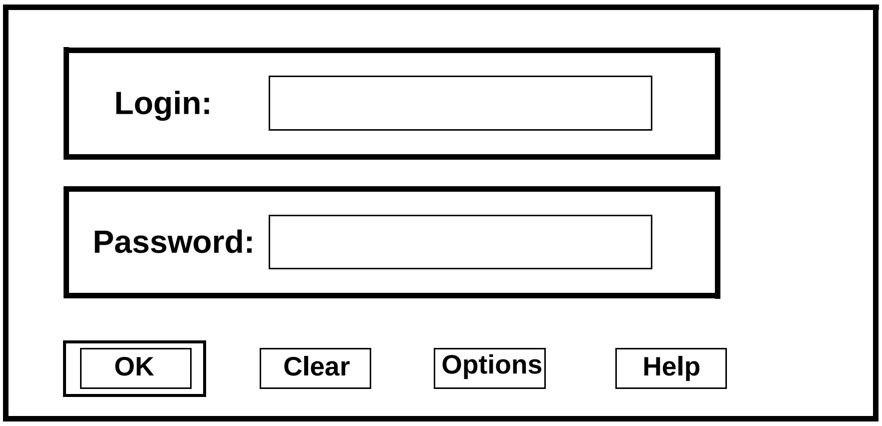


Figure 3-1. HP-UX Log In Dialog Box

1. Enter the appropriate HP-UX user name in the Login field.
2. Select the Password box (either press <**Tab**> or by point and click with the mouse).
3. Enter the appropriate HP-UX password, and then click on **OK** (or press <**Return**>).

By login as a master, mod or cape user the database will be started automatically and an Icon for STB start-up is displayed.

The HP VUE Front Panel with the main icon bar is displayed.

3.1.1.1 Changes in the HP-VUE environment compared with the HP-VUE standard

Changes in the Front Panel:

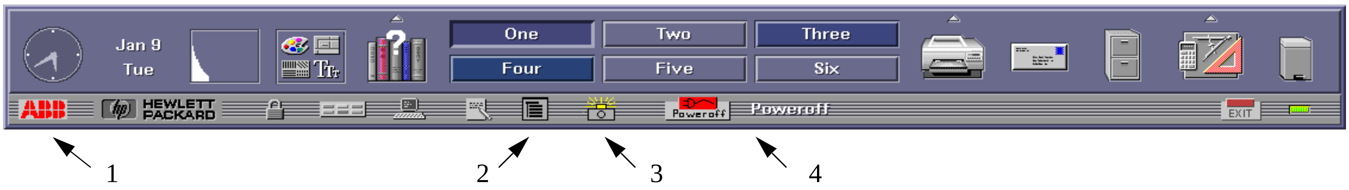


Figure 3-2. HP-VUE Front Panel

1. ABB-Logo
2. Start Icon for STB
3. Hardcopy Icon
4. Poweroff Icon (see also Chapter 2.4.1 for Power Down Engineering Station)

This changes are valid for CAPE and MASTER users. For ROOT user they are valid except the start Icon for STB.

Additional AS500ES menu items above the standard VUE-Workspace menu items

This menu items are activated by pressing the right mouse button

Additional AS500ES menu items for CAPE Users:

- **STB** (start from STB)
- **DB Start** (start of database)
- **DB-Shutdown** (shut down database)
- **DB-State** (show database state in a hpterm)
- **Printer State** (show printer state in a hpterm)
- **New Console Window** (create a new Console Window)

- **ORA-FREE** (show size and free space of all databases in a hpterm)
- **AS 500ES Environment** (show main environment variables in a hpterm)
- **Free Disk Blocks** (show used and free disk space in a hpterm).

Additional AS500ES menu items for MASTER Users:

- **STB** (start from STB)
- **DB Start** (start of database)
- **DB-Shutdown** (shut down database)
- **DB-State** (show database state in a hpterm)
- **Printer State** (show printer state in a hpterm)
- **New Console Window** (create a new Console Window)
- **Boot RTA-Board** (boot of RTA-Board)
- **Free Disk Blocks** (show used and free disk space in a hpterm).

Additional AS500ES menu items for ROOT User:

- **Printer State** (show printer state in a hpterm)
- **New Console Window** (create a new Console Window)
- **Free Disk Blocks** (show used and free disk space in a hpterm
- **Top** (start top command, Display top processes)
- **Glance** (start glance command, System performance monitor)
- **System Swap Info** (start swapinfo command, System swap space information)
- **SETUP**
- **SETUP -C** (CHECK MODE)
- **SETUP -D** (DATE)
- **SETUP -U** (REINSTALL)
- **SETUP -S** (RESET)
- **SETUP -X** (X STATIONS)
- **System Backup** (start asb_tar script).

3.1.2 Logging In to a Specific Project or the Administration Function

To log in to a specific AdvaBuild project:

1. Double click the Icon for STB start-up.

This opens the Structure Builder Window, Figure 3-3.

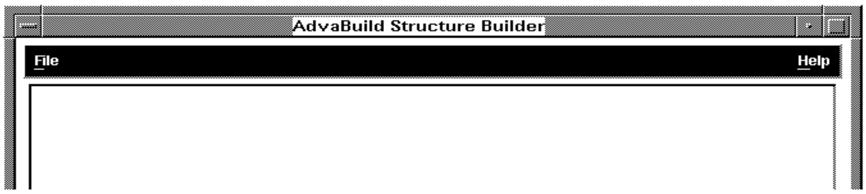


Figure 3-3. Structure Builder Window

2. Choose **Open** from the **File** menu.

   This displays the project log in dialog box, Figure 3-4. This dialog box provides the means for logging in to either a specific project or the Project Administration function.

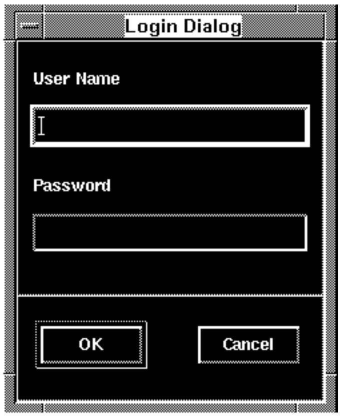


Figure 3-4. Project Log In Dialog Box

   To log in to a specific project, enter the project-specific user name and password that were established when the project was created. To log in to the Project Administration function, you must have logged in to HP-UX with the administration-level user name and password. If you have, then enter the Administrator user name and password in this dialog box.

3. Click inside the User name field and enter the appropriate user name.
4. Click inside the Password field, enter the password, and then click on **OK** (or press <**Return**>).

   When you log into a project, the Structure Builder displays the database tree presentation for the project.

   When you log into the Administration function, the Administration window is displayed. This window provides a menu bar with an **Administer** menu. This menu provides two menu items: **Projects** and **Users**. Selecting one of these menu items opens the corresponding window for working with projects or users.

3.2 Tutorial Master Applications

This tutorial is intended for beginning users of AdvaBuild basic functions. This tutorial leads the uninitiated user step-by-step through some basic application procedures for AdvaBuild functions. The tutorial is intended to give you some quick hands-on experience and familiarize you with the tools and methods for database configuration/documentation.

Keep in mind that this tutorial is only an introduction. AdvaBuild is a powerful and flexible software package that often provides you with more than one method for accomplishing your application building tasks. To keep the tutorial short and simple, just one method is presented here. You should consult Section 3.4, Application Procedures and Section 3.5, Menus for more comprehensive and detailed information on application procedures.

This tutorial is designed to cover each basic procedure in detail just once rather than repeat the same step-by-step instructions over and over. For instance, in the course of this tutorial, you will be instructed to *open* an object on many occasions. The procedure for opening an object is covered in detail just once. Thereafter, you will be expected to know the procedure, or refer to the dedicated section for help.

3.2.1 Overview

This tutorial uses the MAS_EXP (Master Example Project) database which is shipped with the Advant Station. This example project is provided so that you can apply AdvaBuild application procedures to an actual database. You can add, delete, and modify data without having to worry about the consequences.

The users listed in Table 3-2 are pre-configured. Project names MAS_STD, MAS_EXP are reserved names and must not be used for any self-created projects.

Table 3-2. Database Users

PROJECT/ APPLICATION	USER ID	PASSWORD
Master Standard Project	MAS_STD_CFG MAS_STD_RDR	MAS_STD_CFG MAS_STD_RDR
Master Example Project	MAS_EXP_CFG, MAS_EXP_RDR	MAS_EXP_CFG, MAS_EXP_RDR
Administration	ADMIN	ADMIN

The basic application procedures covered in this tutorial are:

- How to navigate the project tree structure
- How to open an object and edit object attributes via the Template Builder
- How to insert an object
- How how to insert and edit objects via the Multiple-Object Template form
- How to move objects from the tree to the list presentation in the Structure Builder
- How to print database documentation packages.

3.2.2 Logging In

To log in to the MAS_EXP database:

1. Log in to HP-UX using the Master database user name and password (Refer to Section 3.1, Application Start-up for details.)
2. When the Login to the database dialog box is displayed, enter **mas_exp_cfg** for user name, and **mas_exp_cfg** again for the password.
3. Press <**Return**>.

   This displays the Project object and its direct children, Figure 3-5.

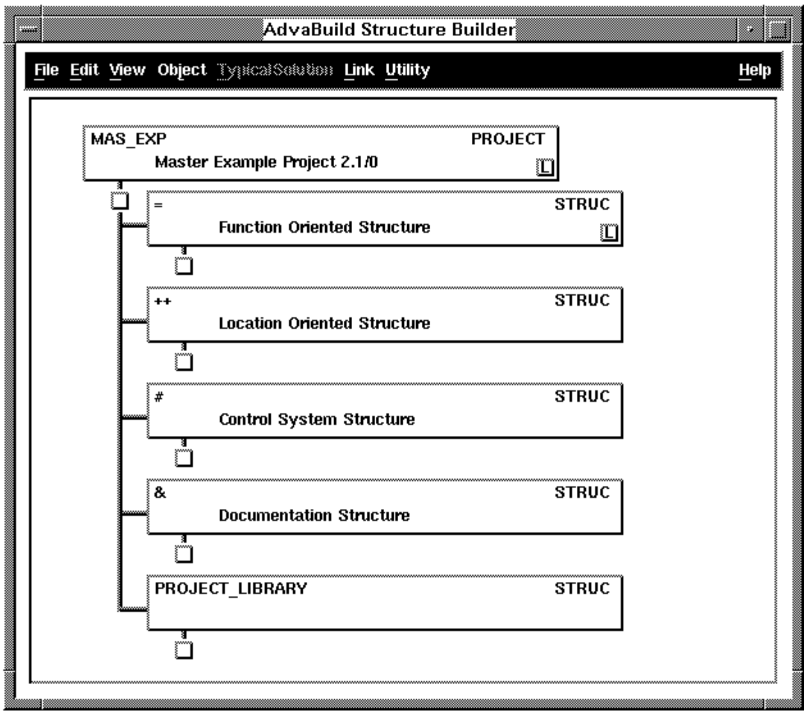


Figure 3-5. mas_exp Project Tree Presentation

The Project object is the root or top level object of the MAS_EXP database. The direct children of the Project object represent the Project Library, the Function-, Location-, Control-, and Documentation Structures of the project database.

All objects have the same basic layout, showing Object ID, Object Type, and an optional comment. A subtree button is provided when additional levels of objects (children) exist under the object, Figure 3-6.

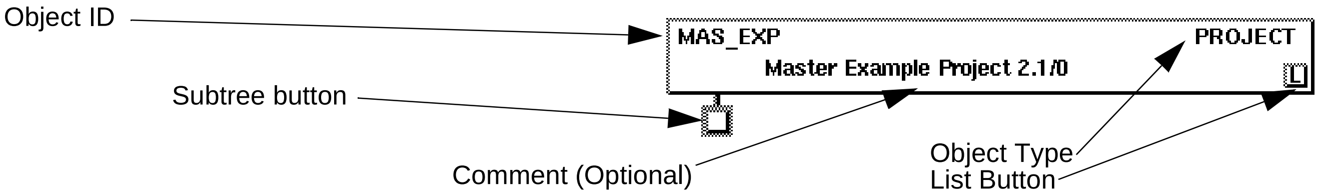


Figure 3-6. Layout of an Object

3.2.3 Navigating the Project Structure

This section describes how to navigate down through the various levels of the database, and how to establish a lower level object as the temporary root for the project.

To expand the tree presentation to show a lower level of objects:

1. Click on the subtree button for the = object of type (STRUC).

   This displays the next level of objects under = Figure 3-7.

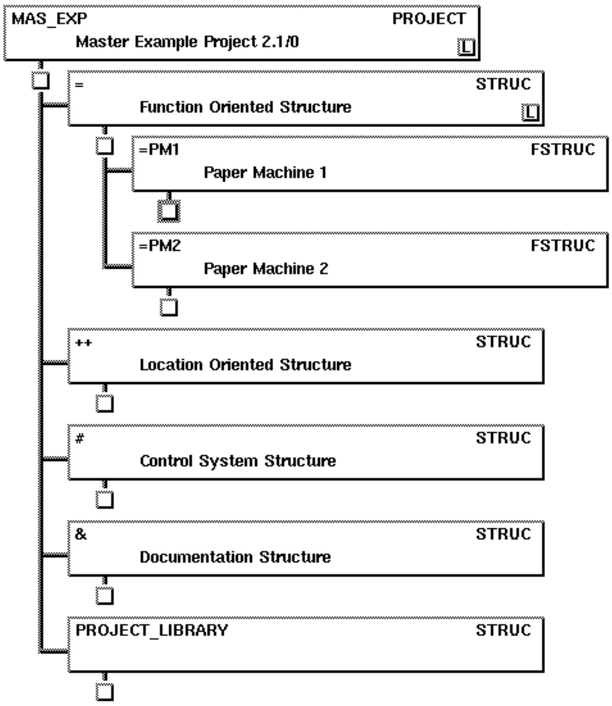


Figure 3-7. Expanding the Database Subtree

To close a subtree, click the activated subtree button again.

NOTE

If you hold down the shift key and then press the subtree button, the entire subtree below = is displayed. This is generally not recommended for any subtree that is very large because it may be very time-consuming (or, in some cases impossible) to display the entire subtree.

2. Click on the subtree button for the =PM1 object.
3. Click on the subtree button for the =PM1DRY object.
   Click on the subtree button for the =PM1DRY19M object.
   Click on the subtree button for the =PM1DRY19MJ0 object.

This displays the =PM1 subtree, Figure 3-8.

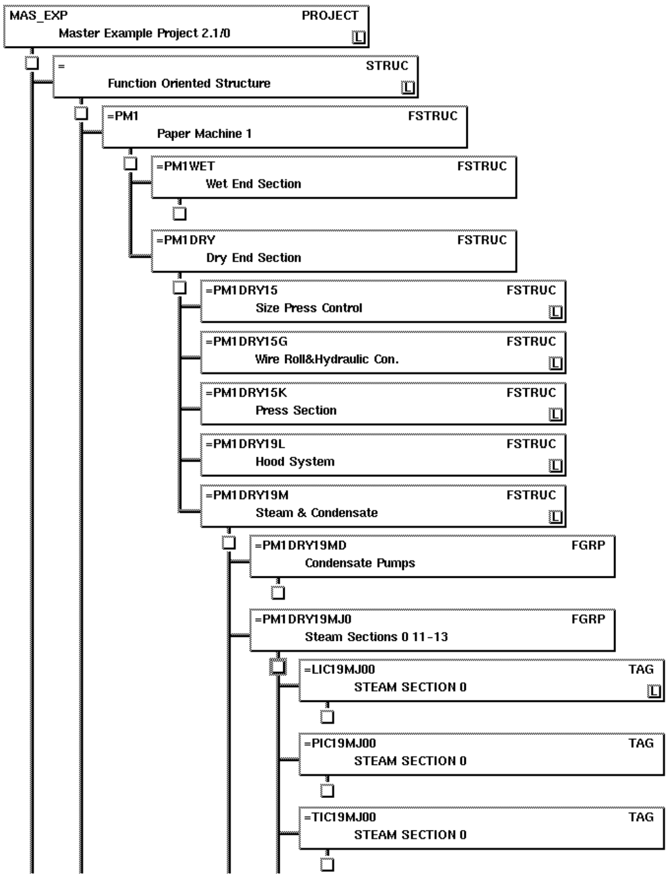


Figure 3-8. =PM1 Subtree

4. Click on the =LIC19MJ00 object, and then choose **Temporary Root** from the **View** menu and then click on the OK button in the appearing window.

   This removes from display all the objects above =LIC19MJ00 and makes =LIC19MJ00 the temporary root object. This facilitates working with the =LIC19MJ00 subtree,

   Notice the Parent button on the left side of the temporary root object. When you press the Parent button, the parent of the temporary root object and its direct children are displayed. This way you can navigate back to the original root (Project) object one-level-at-a-time. Or you can choose **Project Root** from the **View** menu to navigate directly to the Project object.

This displays the =LIC19MJ00 object as temporary root, Figure 3-9.

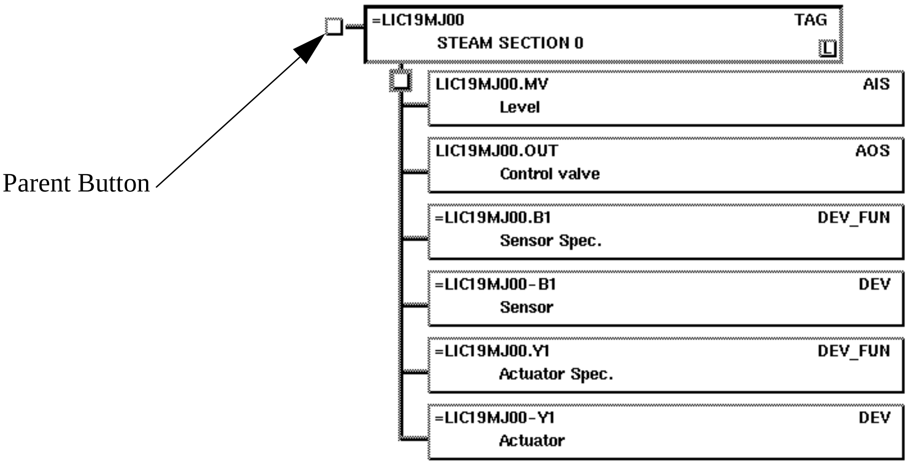

Parent Button

Figure 3-9. =LIC19MJ00 as Temporary Root Object

3.2.4 Using the Template Builder

The Template Builder provides the means to view and edit object attributes via fill-in templates. This section describes how to invoke the Template Builder for a selected object, and how to define object attributes in the Template Builder.

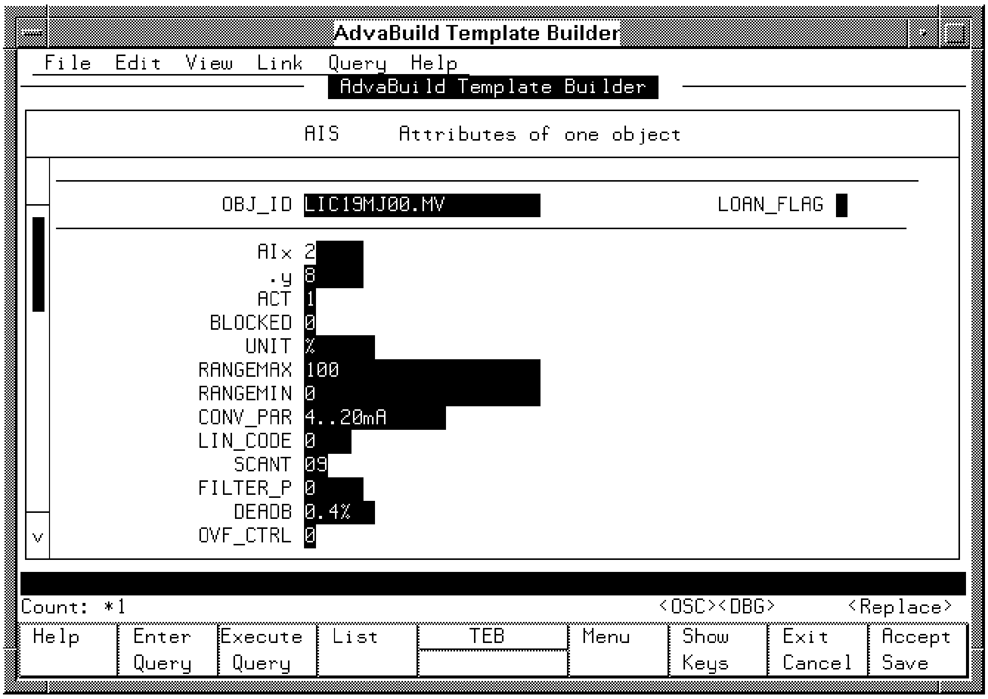
3.2.4.1 Opening an Object and Invoking the Template Builder

To open the LIC19MJ00.MV object and invoke the Template Builder:

1. Either double-click on the LIC19MJ00.MV object, or click once and then choose **Open** from the **Object** menu. Either method can be used to open an object.

   When you open this object, the Single-object form for LIC19MJ00.MV is displayed, Figure 3-10. This form displays just one object at a time. A multiple-object form which displays multiple objects of a selected type in a selected subtree is described later in this tutorial.

The cursor is initially located on the first configurable field. Its position is indicated by highlighting the character as shown in Figure 3-10.



AIS Attributes of one object

OBJ_ID LIC19MJ00.MV
LOAN_FLAG
AIx 2
.y 8
ACT 1
BLOCKED 0
UNIT %
RANGEMAX 100
RANGEMIN 0
CONV_PAR 4..20mA
LIN_CODE 0
SCANT 09
FILTER_P 0
DEADB 0.4%
OVF_CTRL 0

Count: *1 <OSC><DBG> <Replace>

Cursor location indicated by a red highlighted character

Figure 3-10. Single-Object Form

The Template Builder uses the Forms user interface which is described in detail in Section 1.9.4, FORMS User Interface. This tutorial provides a brief introduction to the Forms user interface.

3.2.4.2 Editing Object Attributes

Some template fields have a pre-defined list of values that you can select from. For others, you need to either enter values manually via the keyboard, or copy a value from another template. This section covers manual entry and list of values. The copy method is covered later under the discussion of the Multiple-object form.

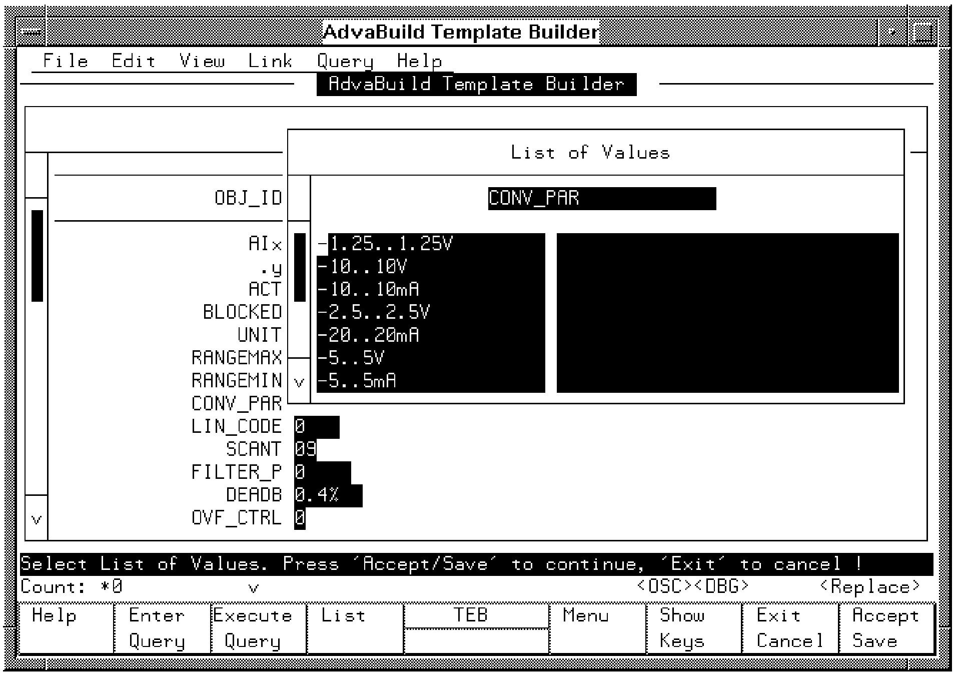
To edit object attributes using manual keyboard entry and List of Values:

1. Navigate to the attribute RANGEMAX.

   You navigate between fields using <**Tab**> (forward) and <**Shift**-**Tab**> (backward).

2. Enter a lower maximum value (for example: **90**).
3. Move the cursor to the CONV_PAR field.
4. Press <**f4**> or click on the **List** bottom line function key.

This opens a List of Values window for CONV_PAR, Figure 3-11.



```
OBJ_ID      CONV_PAR
AIx         -1.25..1.25V
.y          -10..10V
ACT         -10..10mA
BLOCKED     -2.5..2.5V
UNIT        -20..20mA
RANGEMAX    -5..5V
RANGEMIN    -5..5mA
CONV_PAR
LIN_CODE    0
SCANT       09
FILTER_P    0
DEADB       0.4%
OVF_CTRL    0

Select List of Values. Press ´Accept/Save´ to continue, ´Exit´ to cancel !
Count: *0            v                         <OSC><DBG>      <Replace>
```

Figure 3-11. Example of List of Values Window

5. Use the down arrow key to highlight **0...10V**, and press <**f8**> (**Accept/Save**).

   This closes the List of Values window and enters **0...10V** in the CONV_PAR field.

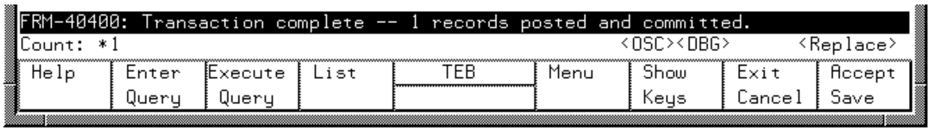
6. You can continue to configure, or save your work and exit this template as described in Section 3.2.4.3, Saving and Exiting an Object.

3.2.4.3 Saving and Exiting an Object

When you work in Template Builder, the work is initially recorded in a temporary workspace. To record the work in the project database and exit the Template Builder:

1. Press <**f8**> (**Accept/Save**).

   When the save is finished, a message is displayed in the message line, Figure 3-12.



```
FRM-40400: Transaction complete -- 1 records posted and committed.
Count: *1                                      <OSC><DBG>      <Replace>
```

Figure 3-12. End of Save Prompt

2. Press <**f7**> (**Exit**).

   This closes the Single-object form.

3.2.5 Inserting an Object

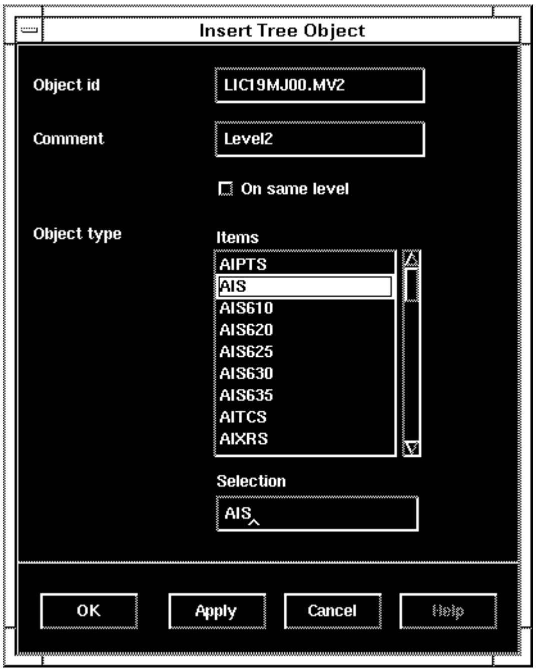
Insert a AIS object in the =LIC19MJ00 subtree to create a new signal. The AIS object is a child of the TAG object in the =LIC19MJ00 subtree. To insert the object:

1. Click on the TAG object at the bottom of the =LIC19MJ00 subtree.

   This establishes the TAG object as the parent for the new AIS object.

2. Choose **Insert** from the **Edit** menu.

   This displays the Insert Object dialog box, Figure 3-13. The selected parent object is indicated in the Object ID field and a list of objects that you can insert as children is provided in the Object Type Items selection box. The On same level button is used when you want to insert an object as a sibling of the selected object. Since the AIS object must be a child of the TAG object in the Function oriented Structure, ignore this button.



Insert Tree Object

Object id: LIC19MJ00.MV2

Comment: Level2

☐ On same level

Object type: Items
- AIPTS
- AIS
- AIS610
- AIS620
- AIS625
- AIS630
- AIS635
- AITCS
- AIXRS

Selection: AIS

Figure 3-13. Insert Dialog Box

3. Click inside the Object ID field, delete the default entry, and then enter: **LIC19MJ00.MV2**.

   You can double-click and press <**backspace**> to delete the entire contents, or use the arrow keys and <**backspace**> to delete a specific character.

4. Click inside the Comment field and enter: **LIC19MJ00.MV2**.

5. Select **AIS** from the Object type selection box.

6. Click on **OK** to accept entries.

   This inserts AIS object **LIC19MJ00.MV2** under the =**LIC19MJ00.** subtree with default attribute values.

3.2.6 Inserting Objects via the Template Builder Multiple-Object Form

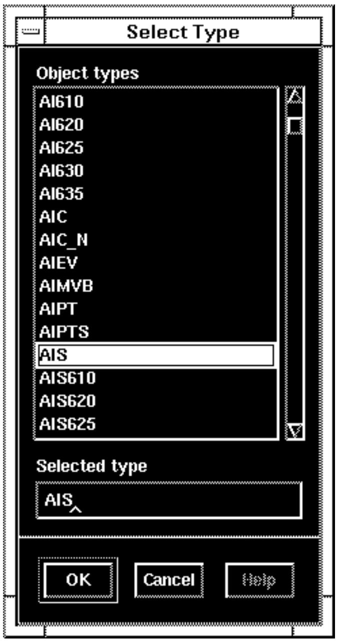
You can view and edit multiple objects of a selected type in a selected subtree via the multiple object form. You can also use this form to insert objects of the selected type. To insert additional AIS objects via the multiple-object form:

1. Click on the =LIC19MJ00 object.

   This establishes the scope of the multiple-object form (that is, the scope is all objects in the =LIC19MJ00 subtree of the type that you select in step 2 below).

2. Choose **Open Type** from the **Object** menu.

   This displays the Object Type selection box,Figure 3-14.



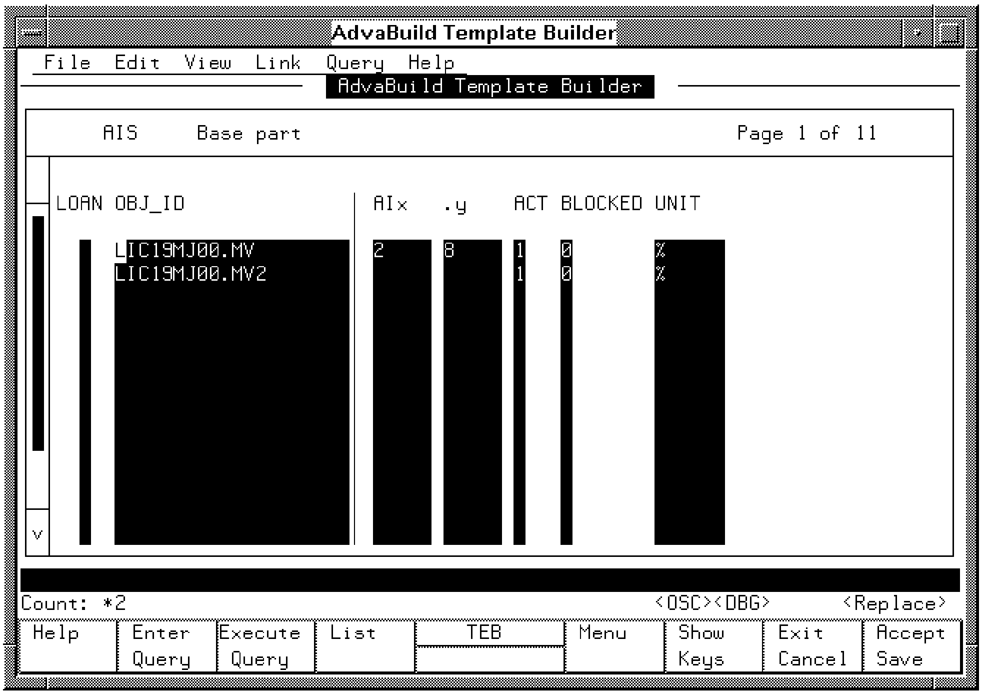
Selected type

AIS

Figure 3-14. Object Type Selection Box

3. Scroll though the list and select **AIS** and then click on **OK**.

   This displays a multiple-object template form for all AIS objects in the =LIC19MJ00 subtree, Figure 3-15. Each object occupies one row. Only a few fields can be shown on the display at a time. You can view other fields by scrolling forward (<**Tab**>) and backward (<**Shift-Tab**>).

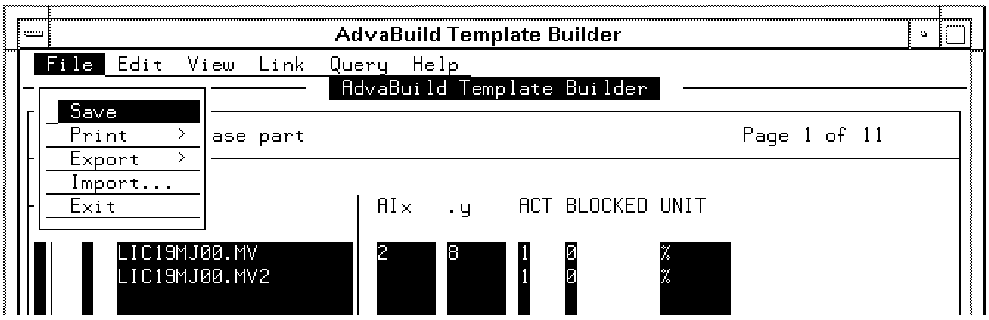


```
LOAN OBJ_ID              AIx   .y    ACT BLOCKED UNIT
     LIC19MJ00.MV        2     8     1   0       %
     LIC19MJ00.MV2                   1   0       %

Count: *2                               <OSC><DBG>    <Replace>
```

Figure 3-15. Multiple-object Form for AIS Objects

4. Select the **Menu** function (<**f5**>) and press **<Return>**.

   This moves the cursor onto the menu bar and pops up the **File** menu, Figure 3-16.


AdvaBuild Template Builder
File Edit View Link Query Help
AdvaBuild Template Builder
Save
Print >
Export >
Import...
Exit
ase part
Page 1 of 11

```
                    AIx   .y    ACT BLOCKED UNIT
LIC19MJ00.MV        2     8     1   0       %
LIC19MJ00.MV2                   1   0       %
```

Figure 3-16. Pop Up Menu in Template Builder

5. Use the right arrow key to move the cursor over the **Edit** menu.

   This displays the **Edit** menu.

6. Use the down arrow key to highlight **Copy** and then press <**Return**> to select it.

This pops up a cascade menu with **Copy** options, Figure 3-17.

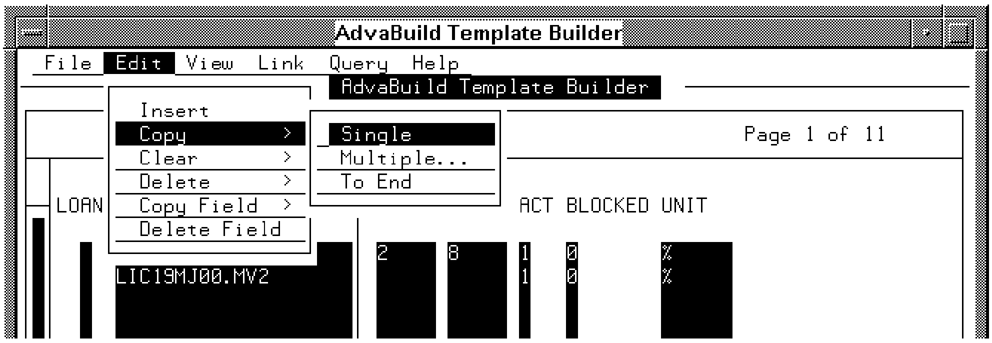
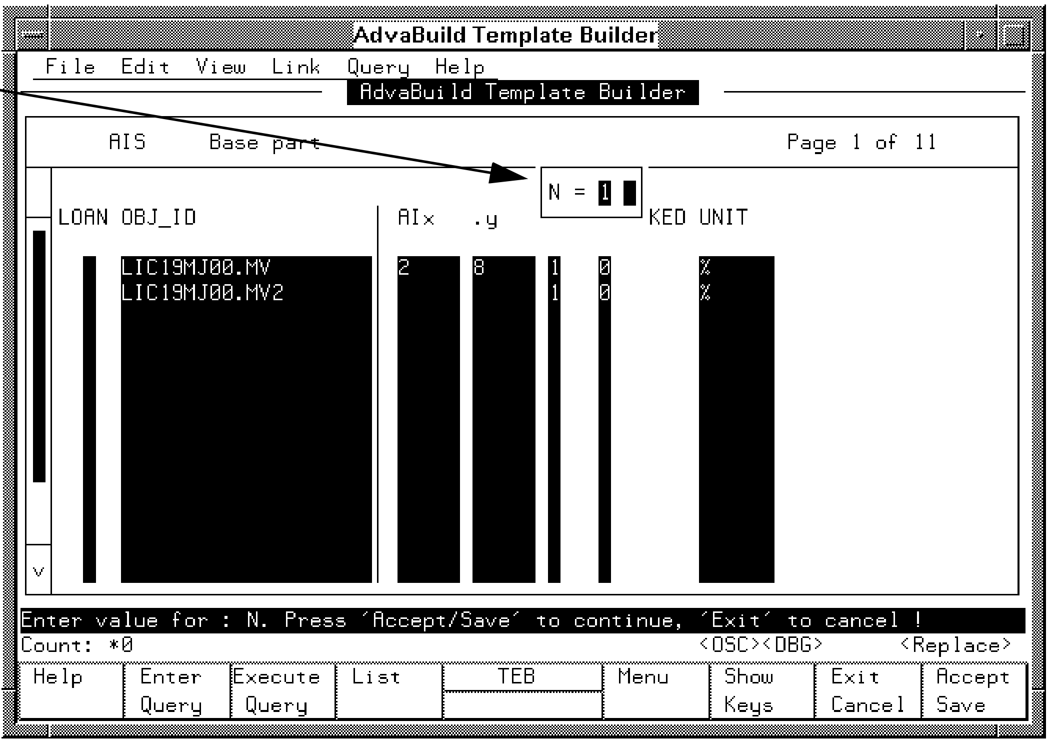


Figure 3-17. Navigating Menus in the Template Builder

7. Use the down arrow key to highlight **Multiple**, and then press <**Return**>.

   This displays a box for entering the number of AIS objects to insert, Figure 3-18.



Enter value for : N. Press 'Accept/Save' to continue, 'Exit' to cancel !

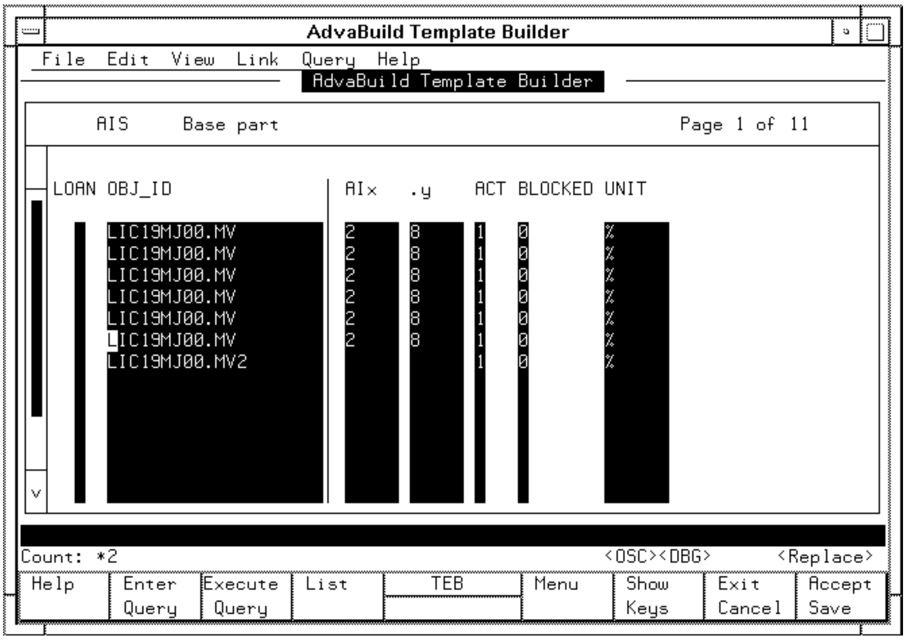
Count: *0

Enter number of objects in this box

Figure 3-18. Dialog for Specifying Number of Objects

8. Enter **5** to insert five new AIS objects, and then press **Accept/Save**.

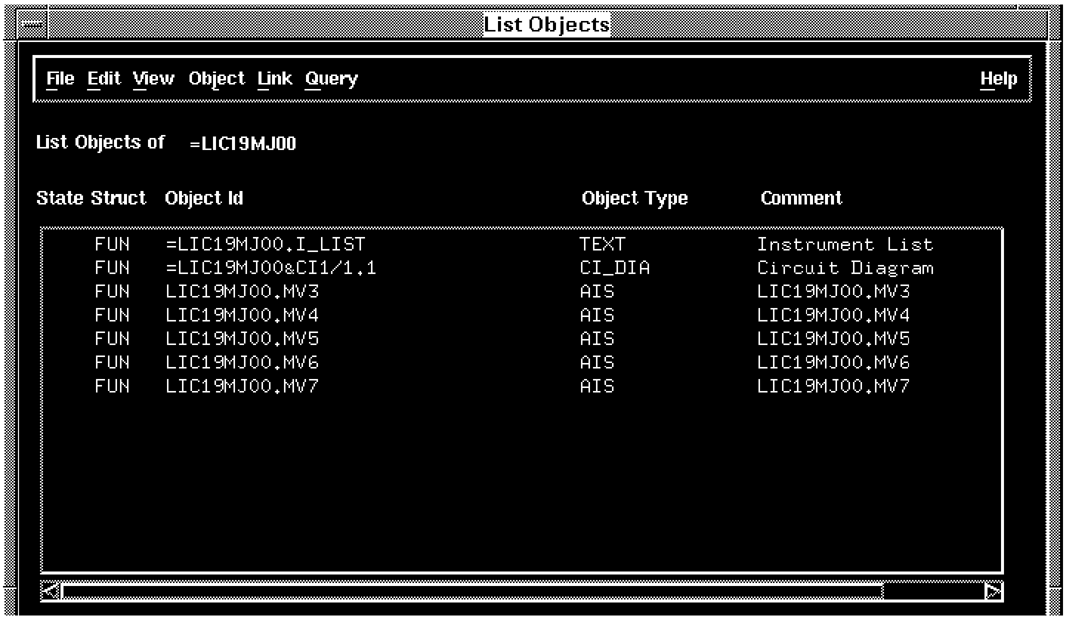
   This creates five new rows with five new AIS objects, Figure 3-18. These new objects have the same attribute values as f=LIC19MJ00.MV, including Object ID.


AdvaBuild Template Builder
File Edit View Link Query Help
AdvaBuild Template Builder
Help
Enter Query
Execute Query
List
TEB
Menu
Show Keys
Exit Cancel
Accept Save

```
   AIS     Base part                               Page 1 of 11

LOAN OBJ_ID              AIx   .y    ACT BLOCKED UNIT
     LIC19MJ00.MV        2     8     1   0       %
     LIC19MJ00.MV        2     8     1   0       %
     LIC19MJ00.MV        2     8     1   0       %
     LIC19MJ00.MV        2     8     1   0       %
     LIC19MJ00.MV        2     8     1   0       %
     LIC19MJ00.MV        2     8     1   0       %
     LIC19MJ00.MV2                   1   0       %

Count: *2                                    <OSC><DBG>     <Replace>
```

Figure 3-19. New AIS Objects Inserted on the Multiple-Object Form

9. Enter a unique object ID for each new AIS object. Use the up, down, left, and right arrow keys to navigate within a field as required.
10. Select **Accept/Save** to commit your entries when you are done, Figure 3-21.
11. To check the successfully insertion, navigate in the Structure Builder to the object =LIC19MJ00 and click on the list button. All list objects for =LIC19MJ00 are displayed, including the new ones, inserted using the Template Builder, Figure 3-20.
    The menu bar in this window provides the same functionality as in the tree representation.



```
List Objects of   =LIC19MJ00

State Struct  Object Id                      Object Type    Comment
     FUN      =LIC19MJ00.I_LIST              TEXT           Instrument List
     FUN      =LIC19MJ00&CI1/1.1             CI_DIA         Circuit Diagram
     FUN      LIC19MJ00.MV3                  AIS            LIC19MJ00.MV3
     FUN      LIC19MJ00.MV4                  AIS            LIC19MJ00.MV4
     FUN      LIC19MJ00.MV5                  AIS            LIC19MJ00.MV5
     FUN      LIC19MJ00.MV6                  AIS            LIC19MJ00.MV6
     FUN      LIC19MJ00.MV7                  AIS            LIC19MJ00.MV7
```

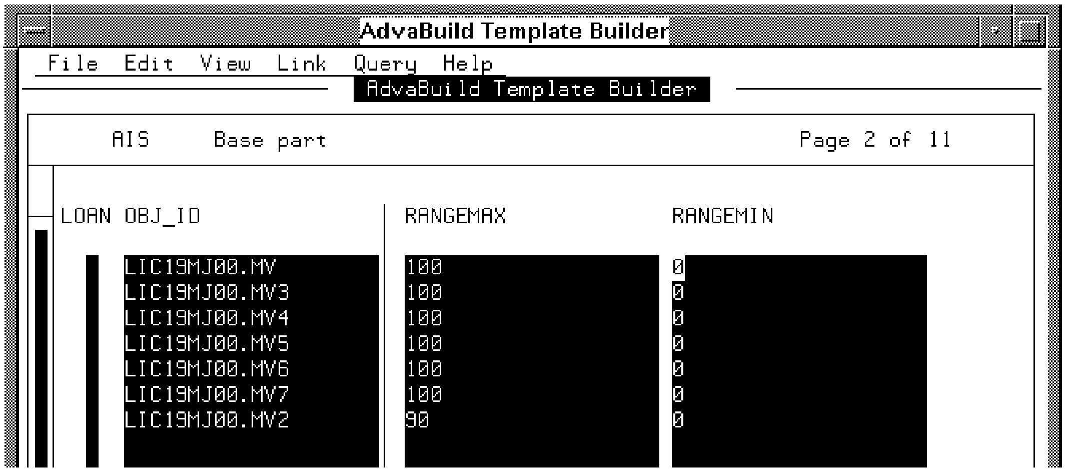
Figure 3-20. Display new AIS Objects STB List Object Window

12. Select Close from the File menu to close this window.

3.2.7 Editing Multiple Objects from the Multiple-object Form

To edit the RAMGEMIN field in all AIS objects in the =LIC19MJ00 subtree:

Navigate to the RAMGEMIN field for =LIC19MJ00.MVPress <**Tab**> to scroll right until the RAMGEMIN field is visible (on page 3 of the template)Figure 3-21.


AdvaBuild Template Builder
File Edit View Link Query Help
AdvaBuild Template Builder

AIS Base part Page 2 of 11

LOAN	OBJ_ID	RANGEMAX	RANGEMIN
	LIC19MJ00.MV	100	0
	LIC19MJ00.MV3	100	0
	LIC19MJ00.MV4	100	0
	LIC19MJ00.MV5	100	0
	LIC19MJ00.MV6	100	0
	LIC19MJ00.MV7	100	0
	LIC19MJ00.MV2	90	0

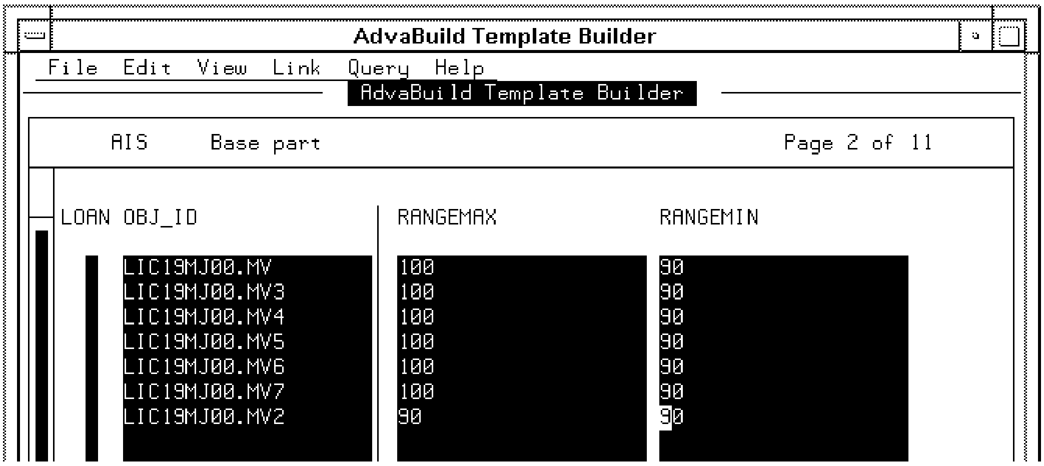
Figure 3-21. TEB Multiple Object Form

13. Enter a new minimum value (for example 90), either by manual keyboard entry or by list of values.
14. Use the down arrow key to navigate to the RANGEMIN field for LIC19MJ00.MV3.
15. Choose **Copy Field** from the **Edit** menu and press <**Return**>.

    This pops up a cascade menu with **Copy Field** options.

16. Choose **To End**.

    This copies the new minimum value from the field above (LIC19MJ00.MV),to all objects below Figure 3-22.


AdvaBuild Template Builder
File Edit View Link Query Help
AdvaBuild Template Builder

AIS Base part Page 2 of 11

LOAN	OBJ_ID	RANGEMAX	RANGEMIN
	LIC19MJ00.MV	100	90
	LIC19MJ00.MV3	100	90
	LIC19MJ00.MV4	100	90
	LIC19MJ00.MV5	100	90
	LIC19MJ00.MV6	100	90
	LIC19MJ00.MV7	100	90
	LIC19MJ00.MV2	90	90

Figure 3-22. Updated RANGEMIN Fields

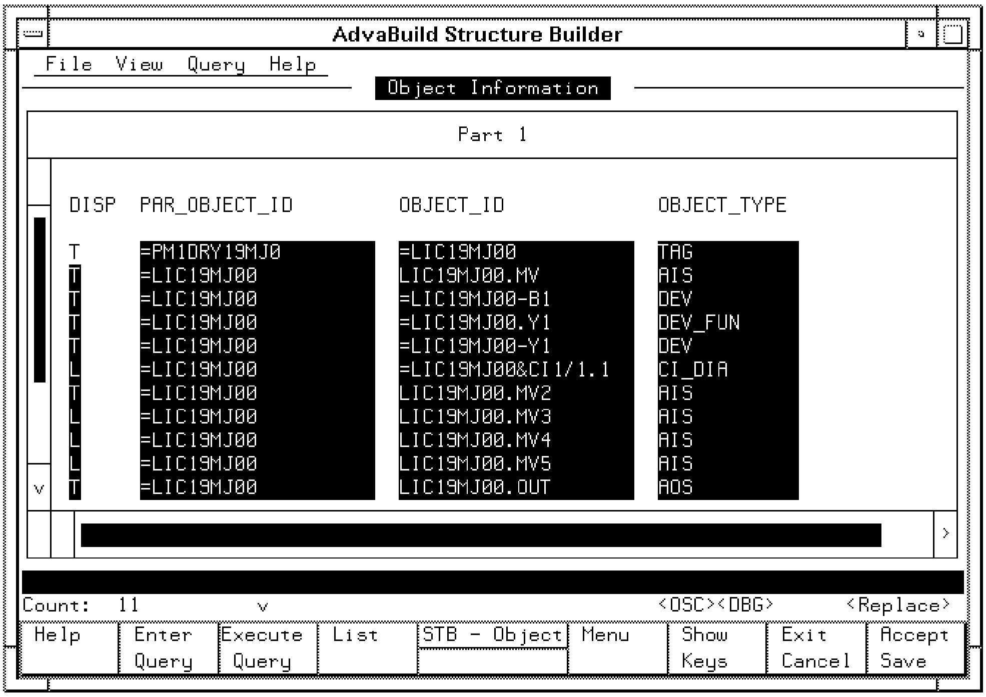
17. Click on **Accept/Save** to save the changes, and then **Exit** to close the Multiple-object form.

3.2.8 Moving Objects from the List to the Tree Presentation

This functionality is supported by the Object Information Window. To put list objects in the tree presentation:

1. Select the =LIC19MJ00 object in the STB.
2. Choose **Information** from the **Object** menu.

   This displays the Object Information window for the =LIC19MJ00 object, Figure 3-23.

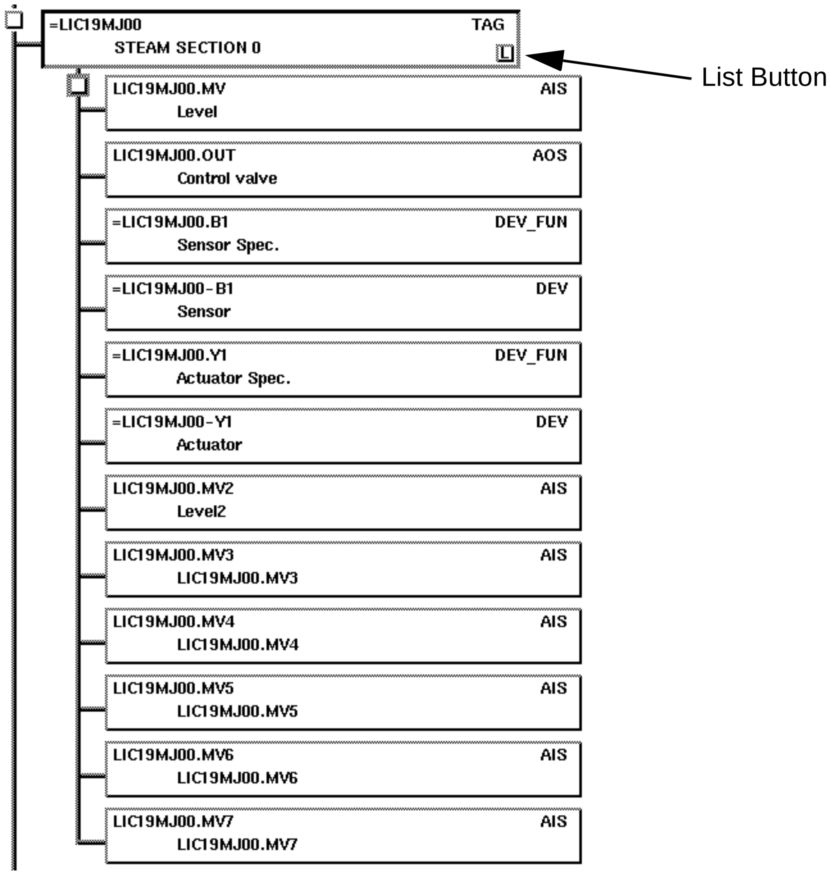


DISP	PAR_OBJECT_ID	OBJECT_ID	OBJECT_TYPE
T	=PM1DRY19MJ0	=LIC19MJ00	TAG
T	=LIC19MJ00	LIC19MJ00.MV	AIS
T	=LIC19MJ00	=LIC19MJ00-B1	DEV
T	=LIC19MJ00	=LIC19MJ00.Y1	DEV_FUN
T	=LIC19MJ00	=LIC19MJ00-Y1	DEV
L	=LIC19MJ00	=LIC19MJ00&CI1/1.1	CI_DIA
T	=LIC19MJ00	LIC19MJ00.MV2	AIS
L	=LIC19MJ00	LIC19MJ00.MV3	AIS
L	=LIC19MJ00	LIC19MJ00.MV4	AIS
L	=LIC19MJ00	LIC19MJ00.MV5	AIS
T	=LIC19MJ00	LIC19MJ00.OUT	AOS

Figure 3-23. Object Information Window

The DISP column indicates the current presentation for each object (**T** = Tree, **L** = List).

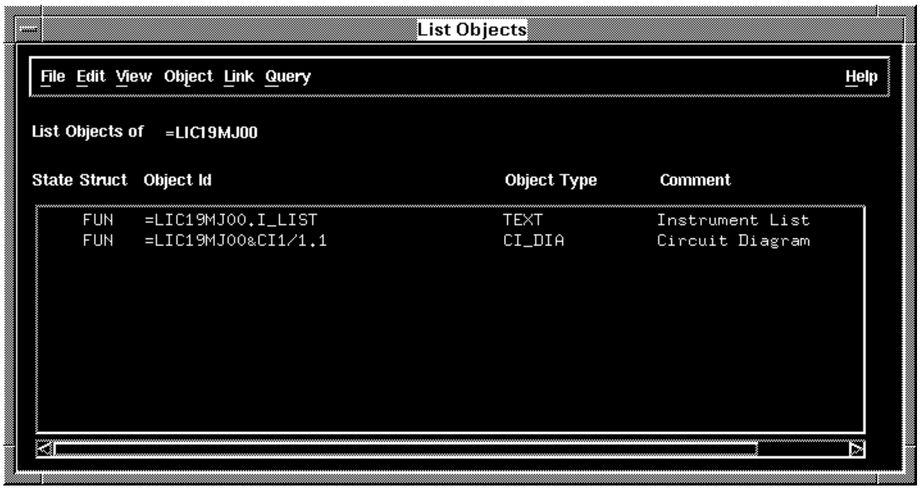
3. Select all AIS objects with =LIC19MJ00 as PAR_OBJECT_ID and change the presentation state from **L** to **T**.
4. When you have changed all the AIS objects, press <**f8**> to accept your entries.
5. Press <**f7**> (**Exit**) to close the Object Information window.
6. Close and open the subtree =LIC19MJ00 using the subtree button. The AIS objects are displayed now as tree objects under =LIC19MJ00.



```
=LIC19MJ00                          TAG
        STEAM SECTION 0
  LIC19MJ00.MV                      AIS
          Level
  LIC19MJ00.OUT                     AOS
          Control valve
  =LIC19MJ00.B1                     DEV_FUN
          Sensor Spec.
  =LIC19MJ00-B1                     DEV
          Sensor
  =LIC19MJ00.Y1                     DEV_FUN
          Actuator Spec.
  =LIC19MJ00-Y1                     DEV
          Actuator
  LIC19MJ00.MV2                     AIS
          Level2
  LIC19MJ00.MV3                     AIS
          LIC19MJ00.MV3
  LIC19MJ00.MV4                     AIS
          LIC19MJ00.MV4
  LIC19MJ00.MV5                     AIS
          LIC19MJ00.MV5
  LIC19MJ00.MV6                     AIS
          LIC19MJ00.MV6
  LIC19MJ00.MV7                     AIS
          LIC19MJ00.MV7
```

Figure 3-24. List Button

7. Click on the list button for =LIC19MJ00.
   The AIS object in the list presentation window are disappeared.



List Objects of =LIC19MJ00

```
State Struct  Object Id                        Object Type    Comment
     FUN      =LIC19MJ00.I_LIST                TEXT           Instrument List
     FUN      =LIC19MJ00&CI1/1.1               CI_DIA         Circuit Diagram
```

Figure 3-25. List Presentation

8. Choose **Close** from the **File** menu to close this window.

3.2.9 Returning to the Project Root

When you are finished editing the LIC19MJ00 subtree, you can restore the tree presentation by choosing **Project Root** from the **View** menu. This displays the Project object and its direct children so that you can navigate to other parts of the project database.

3.2.10 Printing a Documentation Package

Document packages are represented as DSTRUC objects in the Document structure of the project database. A document package contains a pre-configured subset of available document types for a specified subject and application. This tutorial describes one method for printing a document package. For instructions on configuring a documentation package, refer to Section 3.4.5, How to Print Plant Documentation.

To print out a document package:

1. Hold down the <Shift> key and press the subtree button in the & object.

   This opens the Documentation structure subtree, Figure 3-26.

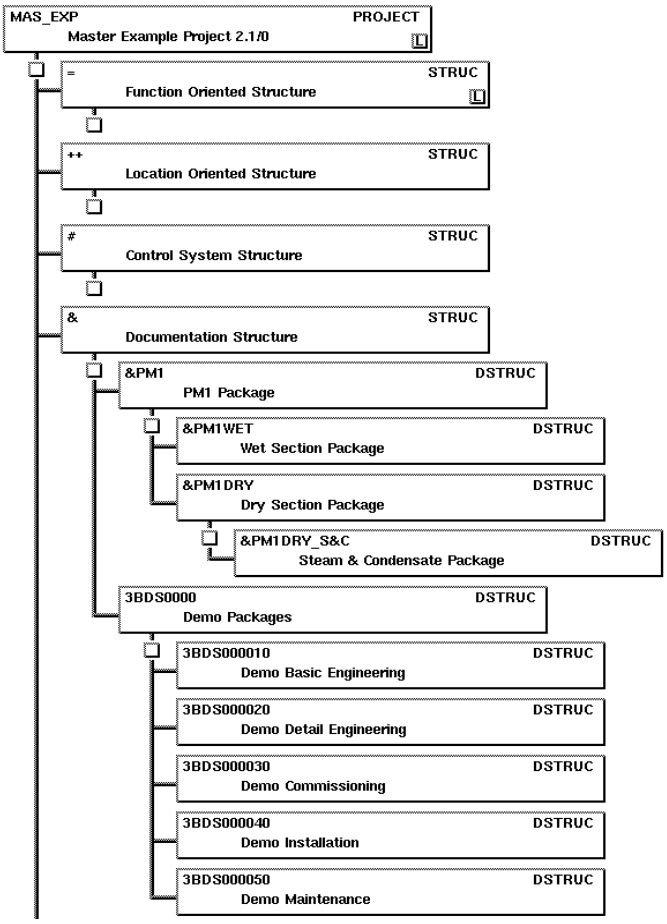


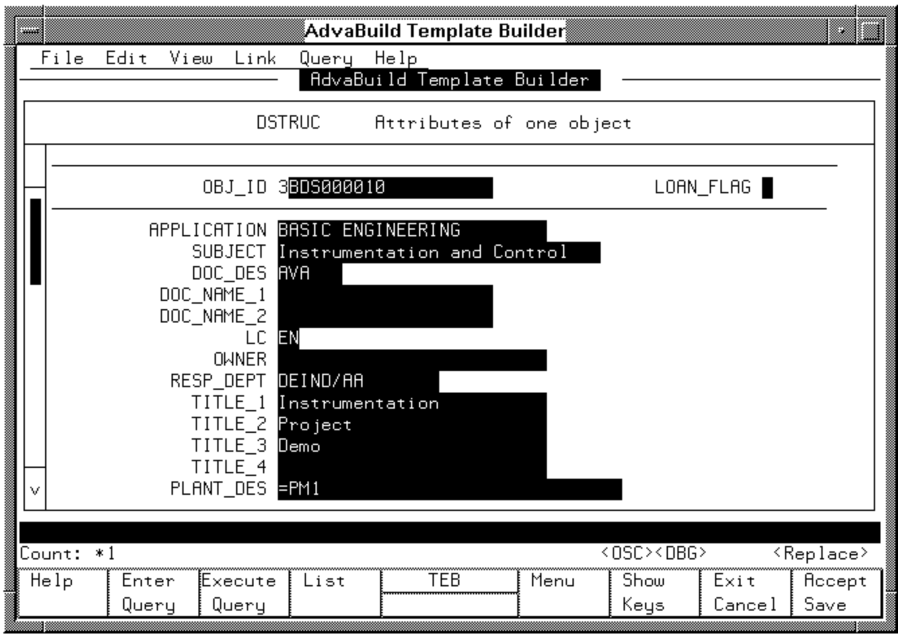
Figure 3-26. Documentation Structure

2. Select object 3BDS000010.

   This object represents a documentation package for a Basic Engineering application. Each document package is configured for a specific documentation application. There are a predefined set of applications, and each application has a predefined set of document types associated with it.

   The application for a documentation package is defined via the Template Builder view for the corresponding DSTRUC object.

Open the Template Builder view for 3BDS000010, Figure 3-27.



```
OBJ_ID 3BDS000010                LOAN_FLAG

APPLICATION BASIC ENGINEERING
    SUBJECT Instrumentation and Control
    DOC_DES AVA
 DOC_NAME_1
 DOC_NAME_2
         LC EN
      OWNER
  RESP_DEPT DEIND/AA
    TITLE_1 Instrumentation
    TITLE_2 Project
    TITLE_3 Demo
    TITLE_4
  PLANT_DES =PM1

Count: *1                               <OSC><DBG>     <Replace>
```

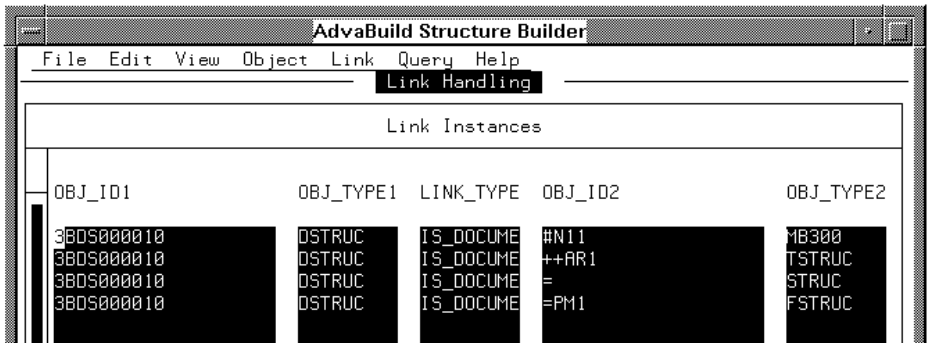
Figure 3-27. Template Builder View of 3BDS000010

   The application is defined in the APPLICATION field. The other fields define contents of the header page for the documentation package.

3. Exit the Template Builder.

4. The scope of this documentation package (that is the set of objects whose documentation is included in the package) is established by linking the DSTRUC object to one or more other objects in the database. 3BDS000010 is linked to the objects #N11, ++AR1, + and +PM1 (in the different structures).To examine the links for this object choose **Of Object** from the **Link** menu. Select the all link types button and accept with **OK** in the appearing window.

This displays the Link Handling window for 3BDS020000, Figure 3-28.

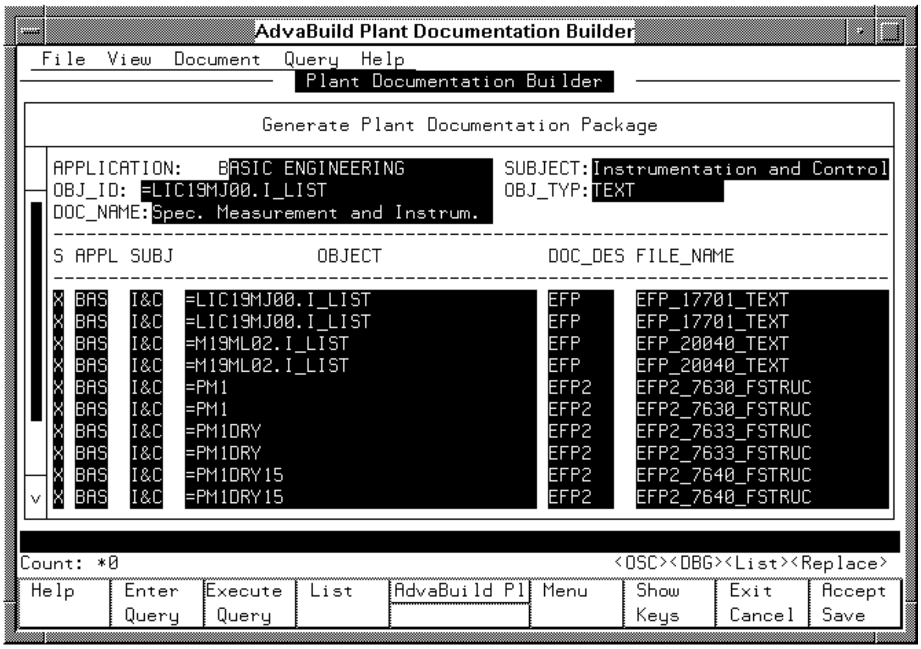

AdvaBuild Structure Builder
File Edit View Object Link Query Help
Link Handling
Link Instances

```
OBJ_ID1              OBJ_TYPE1  LINK_TYPE  OBJ_ID2             OBJ_TYPE2

3BDS000010           DSTRUC     IS_DOCUME  #N11                MB300
3BDS000010           DSTRUC     IS_DOCUME  ++AR1               TSTRUC
3BDS000010           DSTRUC     IS_DOCUME  =                   STRUC
3BDS000010           DSTRUC     IS_DOCUME  =PM1                FSTRUC
```

Figure 3-28. Link Handling Window

5. Exit this window.
6. Choose **Plant Documentation Package** from the **Utility** menu.

   This displays the PDB_Package_Generation window, Figure 3-29.


AdvaBuild Plant Documentation Builder
File View Document Query Help
Plant Documentation Builder
Generate Plant Documentation Package
Help
Enter Query
Execute Query
List
AdvaBuild Pl
Menu
Show Keys
Exit Cancel
Accept Save

```
APPLICATION:   BASIC ENGINEERING              SUBJECT:Instrumentation and Control
OBJ_ID:  =LIC19MJ00.I_LIST                    OBJ_TYP:TEXT
DOC_NAME:Spec. Measurement and Instrum.
---------------------------------------------------------------------------
S APPL SUBJ             OBJECT                DOC_DES FILE_NAME
---------------------------------------------------------------------------
X BAS  I&C  =LIC19MJ00.I_LIST                 EFP     EFP_17701_TEXT
X BAS  I&C  =LIC19MJ00.I_LIST                 EFP     EFP_17701_TEXT
X BAS  I&C  =M19ML02.I_LIST                   EFP     EFP_20040_TEXT
X BAS  I&C  =M19ML02.I_LIST                   EFP     EFP_20040_TEXT
X BAS  I&C  =PM1                              EFP2    EFP2_7630_FSTRUC
X BAS  I&C  =PM1                              EFP2    EFP2_7630_FSTRUC
X BAS  I&C  =PM1DRY                           EFP2    EFP2_7633_FSTRUC
X BAS  I&C  =PM1DRY                           EFP2    EFP2_7633_FSTRUC
X BAS  I&C  =PM1DRY15                         EFP2    EFP2_7640_FSTRUC
X BAS  I&C  =PM1DRY15                         EFP2    EFP2_7640_FSTRUC

Count: *0                                      <OSC><DBG><List><Replace>
```

Figure 3-29. PDB_Package_Generation Window

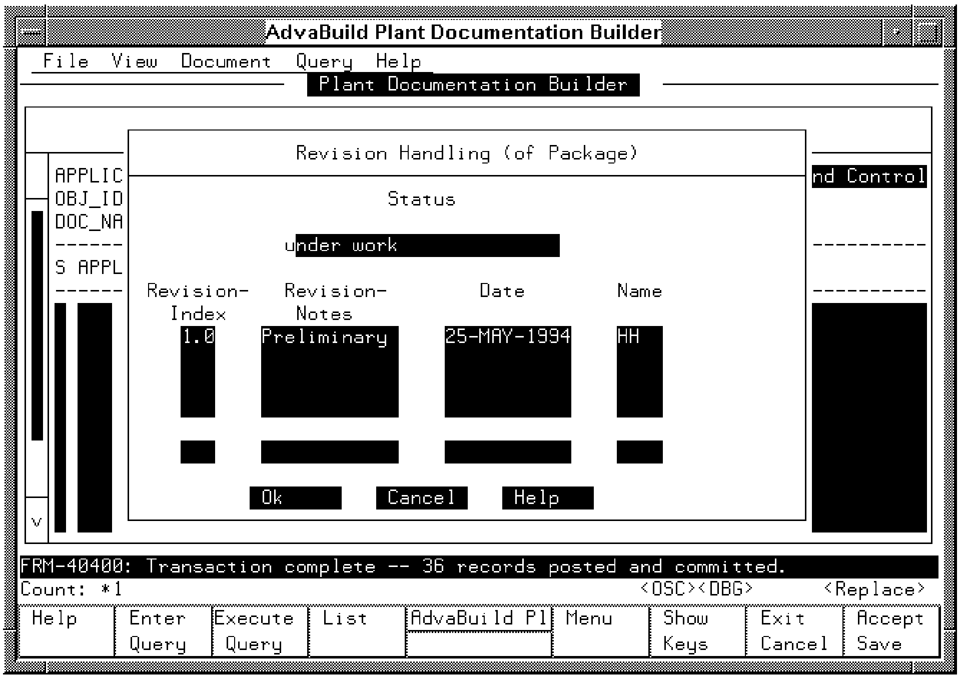
This window shows the contents of this documentation package, based on the selected application and the links that have been established for the DSTRUC object. Initially, this window includes all document types for the specified application. You can limit the printout to just the documents for a specific document type or any combination of documents as needed to meet your documentation requirements.

The print status of each document is indicated in the select (S) column. The default status for all documents is *Selected* as indicated by an **X** in the select column. You can deselect any document by moving the cursor to the line for that document and pressing the spacebar. This deletes the **X**. However this may be too time consuming if you only want to print a few documents. In this case it is easier to deselect all documents and then select just the ones that you want to print.

7. Press <**f5**> to move the cursor into the menu bar.
8. Choose **Deselect All** from the **Document** menu.

   This deselects all documents and moves the cursor to the line for the first document in the package. Now you can select just the documents that you want to print at this time.
9. Scroll through the list of documents an enter an **X** for the =PM1 and =PM1DRY19M&PI1/1.1 documents.
10. When you are finished selecting the documents, choose **Print** from the **Document** menu.

    This displays the Revision Handling dialog box, which provides the means to enter revision control information for your document Figure 3-30.



Revision Handling (of Package)

Status

under work

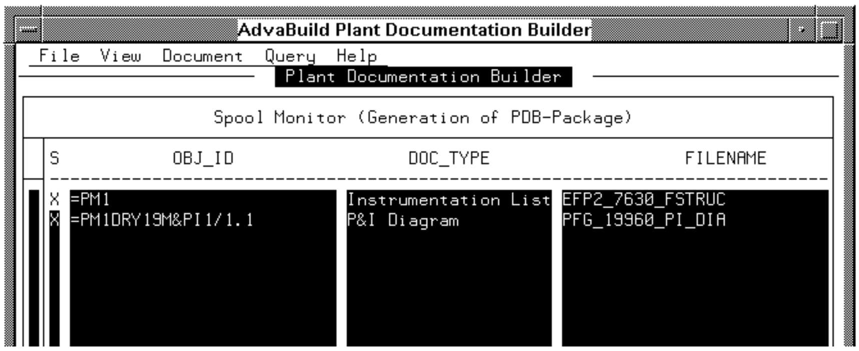
Revision-Index	Revision-Notes	Date	Name
1.0	Preliminary	25-MAY-1994	HH

```
FRM-40400: Transaction complete -- 36 records posted and committed.
Count: *1                                         <OSC><DBG>      <Replace>
```

Figure 3-30. Revision Handling Dialog Box

11. Enter any information as required. Use <**Tab**> to navigate from field to field.
12. To accept the entries and close the revision handling window, press <**f8**> (**accept/save**)when the cursor is located on the **OK** field.

This displays the Print Spool window which provides the means to view the documents that you have selected for printing and gives you one more chance to deselect any if required, Figure 3-31.


AdvaBuild Plant Documentation Builder
File
View
Document
Query
Help
Plant Documentation Builder

Spool Monitor (Generation of PDB-Package)

S	OBJ_ID	DOC_TYPE	FILENAME
X	=PM1	Instrumentation List	EFP2_7630_FSTRUC
X	=PM1DRY19M&PI1/1.1	P&I Diagram	PFG_19960_PI_DIA

Figure 3-31. PDB_Package_Generation Window w/Final List of Documents

13. Press <**f8**> (**Accept/Save**) again to print the document.

This concludes part 1 of the tutorial.

3.3 On-line Builder Tutorial

The On-line Builder may be used to communicate via the MasterBus 300 to MasterPiece 200,Master Batch Controller 200, MasterView 800, MasterGate 230, AC410 and AC450 stations. Those programs may be tested and debugged with the On-line Builder, too.

Using the On-line Builder, you may

- convert DB source code files, save and restore memory dumps
  for objects of type:
  MP200, PCPGM, MV810, MV830, MV850, MG230 and MBC200
  AC410C, AC450C,
  AS520IMS, AS520OS, AS520ES, AS515IMS, AS515OS, AS515ES and AS510 OS.
- convert PC source code files
  for objects of MP200, AC410C, AC450C and PCPGM type.
- use the Commands (like DIMDB, TRDBS, TRPCS).

3.3.1 Overview

This On-line Builder tutorial guides a beginner through the many tools of an Advant Station 500 series Engineering Station. It covers not only On-line Builder but also basic AdvaBuild functions to have a complete reproducible example. However, if you are a experienced AdvaBuild/ FCB user, we suggest to skip the preparation part and directly practice the On-line part with an own PC program as described in {section 4.3.3}.

In this exercise, an application program for a MasterPiece 200 station shall be down-loaded from the Advant Station 500 Engineering Station to the target machine. The application program consists of a DB part and a PC part.

The application program shall be tested and modified. The modified program shall be up-loaded from the target machine into the Engineering Station, thus having an update of the modified program on the Engineering Station, too.

You are invited to do this exercise on your Engineering Station in order to get the first practice with the On-line Builder.

The tutorial consists of two parts:

- Preparations
  - Insert another MP200 object underneath a MB300 object in the DCS structure
  - Create DB elements for the MasterPiece 200/1 station
  - Board type definition for DI and DO boards
  - Link the input and output signals to the DI and DO board
  - Create a small PC program.
- Using the On-line Builder
  - Starting the On-line Builder
  - Copy Source Code Files for Down-load
  - Establish Connection to Target Node
  - Down-load Program to Target Node
  - Test and modify the Application Program
  - Up-load Program from Target Node
  - Disconnect from Target Node
  - Copy Source Code Files for Backtranslate
  - Leave On-line Builder
  - Integrate modified Application Program into Project Structure.

The first tasks are necessary in order to create the environment for using On-line functions. The last tasks cover the On-line functions itself.

3.3.2 Preparations

The following preparations should have been done in advance (not part of the exercise)

- Real-time accelerator board installed in the Engineering Station.
- Real-time accelerator board configured, including network and node number setting.
- Real-time accelerator board connected to MasterBus 300.
- MasterPiece 200/1 station with software QMP280 (V4.0) connected to the same MasterBus 300.
- MasterPiece 200/1 station up and running, operating mode, indicated by P1.
- Engineering Station up and running.

- There is an example project with the name 'MAS_EXP' on the Engineering Station with at least 5 MByte free space.
- The database has been started, the Structure Builder has been started and the project MAS_EXP has been opened.

You will see the Structure Builder window on the screen (see Figure 3-32).

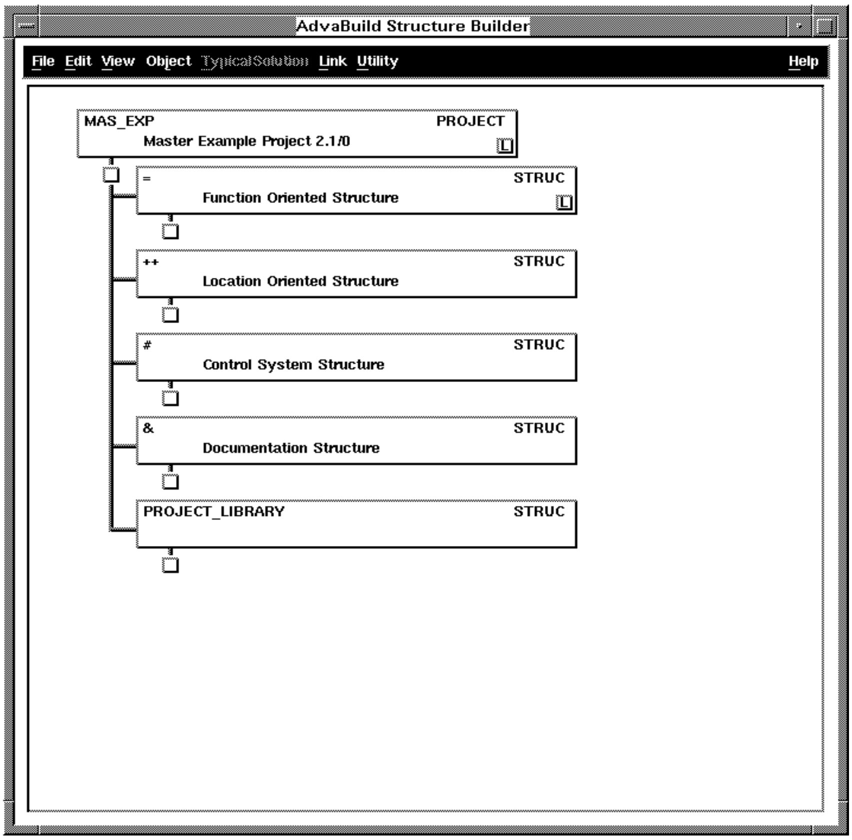

AdvaBuild Structure Builder
File Edit View Object TypicalSolution Link Utility Help
MAS_EXP PROJECT
Master Example Project 2.1/0
= STRUC
Function Oriented Structure
++ STRUC
Location Oriented Structure
# STRUC
Control System Structure
& STRUC
Documentation Structure
PROJECT_LIBRARY STRUC

Figure 3-32. Basic Project Structure 'MAS_EXP'

Insert another MP200 object underneath the MB300 object in the Control System Structure

You will insert a new MP200 object and define this object's attributes. So you don't have to modify an existing MP200 object in the example project MAS_EXP. This MP200 object will be used for the On-line Builder tutorial.

Open the substructure #N11 within the Control System Structure:

- Click on the small square below the object labelled # (the Control System Structure) to open the substructure.
- Click on the small square below the object labelled #N11 to open the next substructure level.

Set the MB300 object named #N11 to temporary root:

- Click on the object labelled **#N11** on a location which is not occupied by text. The appearance of the rectangle will change, looking as if it is pressed down.
- Click on **View** (in the menu bar at the top of the Structure Builder window).
- Click on **Temporary Root** (in the pulldown menu of View) and press the **OK** button.

If there is already an existing object #N11_OnbExa from a tutorial session before, please delete:

- Check, if there is an object labelled #N11_OnbExa in the structure below #N11.
- If it is, click on that object(select), click on **Edit** in the menu bar, then click on **Delete** in the Edit menu. Click on the **Delete** button in the dialogue box.
- After some seconds, your Structure Builder window should look similar to Figure 3-33.

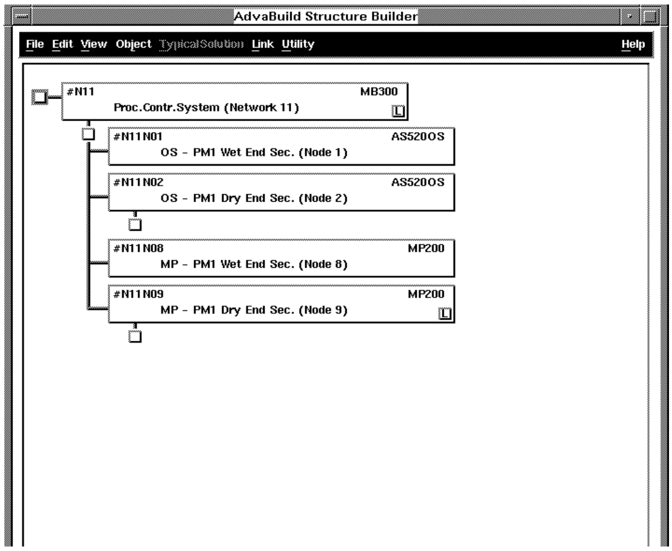

AdvaBuild Structure Builder
File
Edit
View
Object
TypicalSolution
Link
Utility
Help
#N11
MB300
Proc.Contr.System (Network 11)
#N11N01
AS520OS
OS - PM1 Wet End Sec. (Node 1)
#N11N02
AS520OS
OS - PM1 Dry End Sec. (Node 2)
#N11N08
MP200
MP - PM1 Wet End Sec. (Node 8)
#N11N09
MP200
MP - PM1 Dry End Sec. (Node 9)

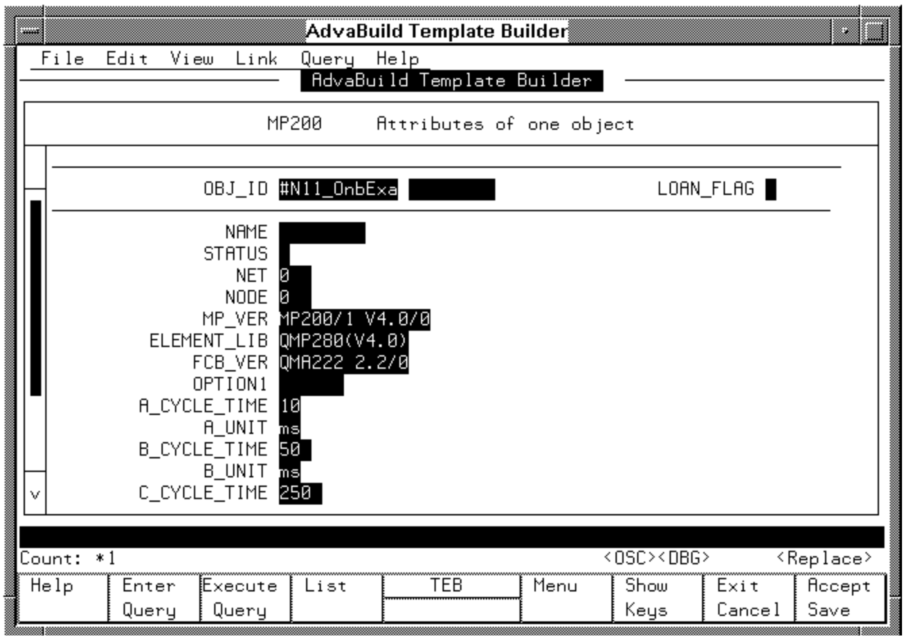
Figure 3-33. DCS substructure #N11

Create a new object #N11_OnbExa of type MP200:

- Click on object **#N11**.
- Click on **Edit** in the menu bar, then click on **Insert** in the Edit menu.
- In the dialogue box, labelled **Insert Tree Object**
  - Click on the field 'Object id', which contains '#N11'.
  - Change the Object ID to **#N11_OnbExa**.
  - Click on the field 'Comment', delete the old text, then type **Testing On-line Builder**.
  - Click on **MP200** in the list of object types to select object type MP200.
  - Click on the **OK** button.
    You will see the new object in the Structure Builder window.

Adjust network and node address of that object using the Template Builder:

- Double click on the new object **#N11_OnbExa**.
- In the dialogue box, labelled **Open Object**
  click on **Template**, then click on the **OK** button.
  You will see the Template Builder window, as in Figure 3-34.



AdvaBuild Template Builder
File Edit View Link Query Help
AdvaBuild Template Builder

MP200 Attributes of one object

OBJ_ID #N11_OnbExa
LOAN_FLAG
NAME
STATUS
NET 0
NODE 0
MP_VER MP200/1 V4.0/0
ELEMENT_LIB QMP200(V4.0)
FCB_VER QMA222 2.2/0
OPTION1
A_CYCLE_TIME 10
A_UNIT ms
B_CYCLE_TIME 50
B_UNIT ms
C_CYCLE_TIME 250

Count: *1 <OSC><DBG> <Replace>

Help
Enter Query
Execute Query
List
TEB
Menu
Show Keys
Exit Cancel
Accept Save

Figure 3-34. Template for MP200 object

- In the Template Builder window
  - Press the **Tab** key on the keyboard, putting the cursor into the text field of **NAME**.
  - Type the name: **ONBEXA**.
  - Press the **Tab** key twice, putting the cursor into the text field of **NE**T.
  - Type the network number of the MasterPiece 200/1 station (e.g. 12).
  - Put the cursor (with the **Tab** key) into the next text field of **NODE**.
  - Type the node number of the MasterPiece 200/1 station (e.g. 2).
  - Press the Tab key three times, putting the cursor into the text field of **FCB_VER**
  - Type in the FCB version: **QMA222 3.1/0** (Use the List of Value - facility).

Remark: You may use the **Tab** key to go down, <**Shift-Tab>** to go up, left arrow key, right arrow key and **Backspace** key to edit within a text field.

  - Click on the **Accept Save** button (in the lower part of the Template Builder window).
  - Leave the Template builder by clicking on the **Exit Cancel** button.

Insert DB elements for the MasterPiece 200/1 station.

You will insert a digital input board, a digital output board, two digital input signals and two digital output signals as DB elements of the MP200.

Open the substructure below #N11:

- Click on the small square below the object #N11.
- As you will see, the MP200 object #N11_OnbExa has a small square inside labelled **L** (the list element button). (See Figure 3-35).

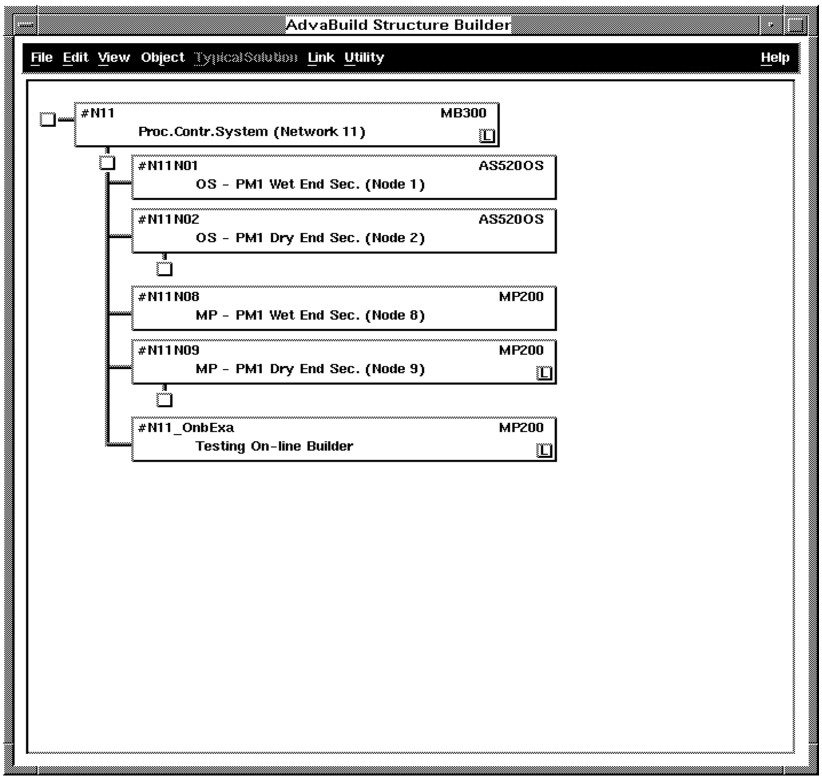
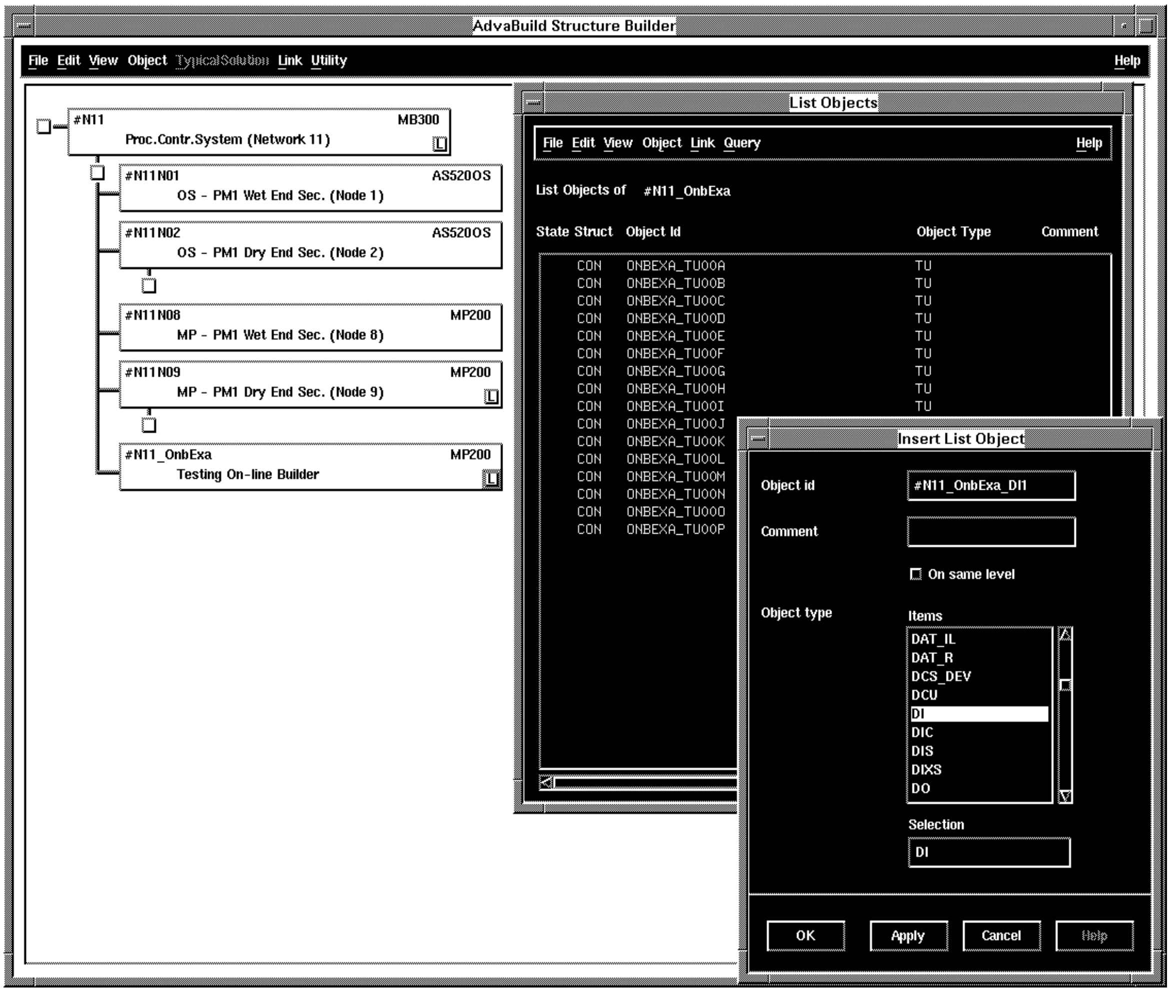

AdvaBuild Structure Builder
File
Edit
View
Object
TypicalSolution
Link
Utility
Help
#N11 MB300
Proc.Contr.System (Network 11)
L
#N11N01 AS520OS
OS - PM1 Wet End Sec. (Node 1)
#N11N02 AS520OS
OS - PM1 Dry End Sec. (Node 2)
#N11N08 MP200
MP - PM1 Wet End Sec. (Node 8)
#N11N09 MP200
MP - PM1 Dry End Sec. (Node 9)
L
#N11_OnbExa MP200
Testing On-line Builder
L

Figure 3-35. Newly inserted object '#N11_OnbExa'

Insert required boards and signal elements as list objects:

- Click on that list element button of the MP200 object #N11_OnbExa.
  You will see a window with a list of DB elements. These DB elements have already been created for the MasterPiece 200/1 station.
- Use the scroll bar on the right hand side of the window to find the end of the list.
- Click on **Edit** in the menu bar of this window, then click on **Insert** in the Edit menu.
  You will see the dialogue box labelled **Insert List Object**.


AdvaBuild Structure Builder
File Edit View Object TypicalSolution Link Utility Help
#N11 MB300
Proc.Contr.System (Network 11)
#N11N01 AS520OS
OS - PM1 Wet End Sec. (Node 1)
#N11N02 AS520OS
OS - PM1 Dry End Sec. (Node 2)
#N11N08 MP200
MP - PM1 Wet End Sec. (Node 8)
#N11N09 MP200
MP - PM1 Dry End Sec. (Node 9)
#N11_OnbExa MP200
Testing On-line Builder
List Objects
File Edit View Object Link Query Help
Insert List Object
OK
Apply
Cancel
Help

List Objects of #N11_OnbExa

State Struct	Object Id	Object Type	Comment
CON	ONBEXA_TU00A	TU	
CON	ONBEXA_TU00B	TU	
CON	ONBEXA_TU00C	TU	
CON	ONBEXA_TU00D	TU	
CON	ONBEXA_TU00E	TU	
CON	ONBEXA_TU00F	TU	
CON	ONBEXA_TU00G	TU	
CON	ONBEXA_TU00H	TU	
CON	ONBEXA_TU00I	TU	
CON	ONBEXA_TU00J		
CON	ONBEXA_TU00K		
CON	ONBEXA_TU00L		
CON	ONBEXA_TU00M		
CON	ONBEXA_TU00N		
CON	ONBEXA_TU00O		
CON	ONBEXA_TU00P		

Object id: #N11_OnbExa_DI1

Comment:

☐ On same level

Object type

Items
- DAT_IL
- DAT_R
- DCS_DEV
- DCU
- DI
- DIC
- DIS
- DIXS
- DO

Selection: DI

Figure 3-36. Insert list objects DI, DO, ...

Within this dialogue box insert:

DI	#N11_OnbExa_DI1
DO	#N11_OnbExa_DO1
DIS	#N11_OnbExa_SWITCH1
DIS	#N11_OnbExa_SWITCH2
DOS	#N11_OnbExa_LAMP1
DOS	#N11_OnbExa_LAMP2

– Click on the Object id field, then change the old text (reading #N11_OnbExa) to **#N11_OnbExa_DI1** (see Figure 3-36).

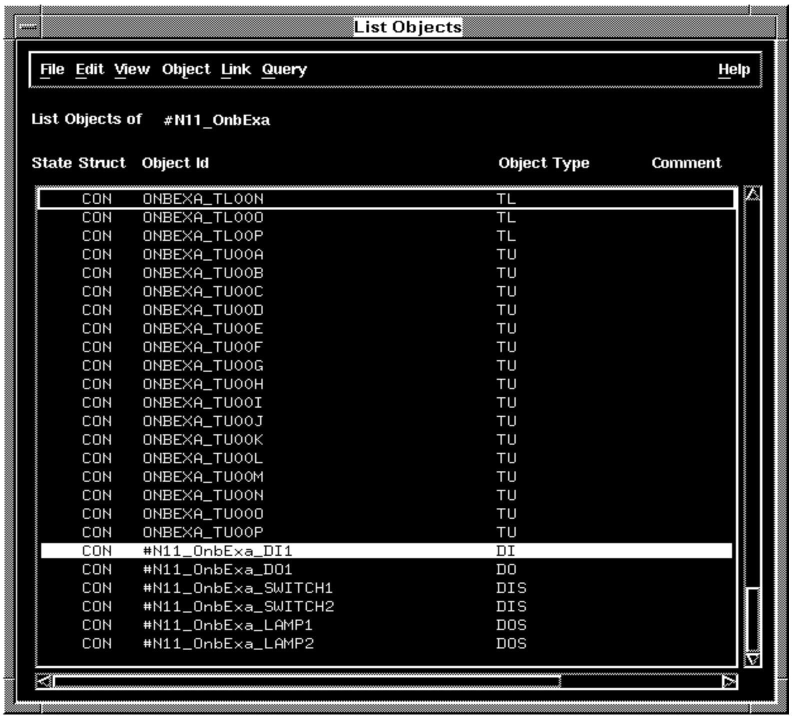
– Click on the second text field Comment and delete the old text.

– Use the scroll bar of Items to find the DB element **DI** within the list of Object types.

– Click on **DI**, then click on the **Apply** button.

- – Again, click on the Object id field, then change the text to **#N11_OnbExa_DO1**.
- – Find **DO** within the list of Object types, click on **DO**, then click on the **Apply** button.
- – Again, click on the Object id field, then change the text to **#N11_OnbExa_SWITCH1**.
- – Find **DIS**, click on **DIS**, then click on the **Apply** button.
- – Again, click on the Object id field, then change the text to **#N11_OnbExa_SWITCH2**.
- – **DIS** is still selected. Click on the **Apply** button.
- – Again, click on the Object id field, then change the text to **#N11_OnbExa_LAMP1**.
- – Find **DOS**, click on **DOS**, then click on the **Apply** button.
- – Again, click on the Object id field, then change the text to **#N11_OnbExa_LAMP2**.
- – **DOS** is still selected. Click on the **Apply** button.
- – All those DB elements you need for the practice have been now created. Leave this dialogue box by clicking on the **Cancel** button.

- Check the list of DB elements to see whether all new DB elements really exist (see Figure 3-37).



List Objects of #N11_OnbExa

State Struct	Object Id	Object Type	Comment
CON	ONBEXA_TLOON	TL	
CON	ONBEXA_TLOOO	TL	
CON	ONBEXA_TLOOP	TL	
CON	ONBEXA_TUOOA	TU	
CON	ONBEXA_TUOOB	TU	
CON	ONBEXA_TUOOC	TU	
CON	ONBEXA_TUOOD	TU	
CON	ONBEXA_TUOOE	TU	
CON	ONBEXA_TUOOF	TU	
CON	ONBEXA_TUOOG	TU	
CON	ONBEXA_TUOOH	TU	
CON	ONBEXA_TUOOI	TU	
CON	ONBEXA_TUOOJ	TU	
CON	ONBEXA_TUOOK	TU	
CON	ONBEXA_TUOOL	TU	
CON	ONBEXA_TUOOM	TU	
CON	ONBEXA_TUOON	TU	
CON	ONBEXA_TUOOO	TU	
CON	ONBEXA_TUOOP	TU	
CON	#N11_OnbExa_DI1	DI	
CON	#N11_OnbExa_DO1	DO	
CON	#N11_OnbExa_SWITCH1	DIS	
CON	#N11_OnbExa_SWITCH2	DIS	
CON	#N11_OnbExa_LAMP1	DOS	
CON	#N11_OnbExa_LAMP2	DOS	

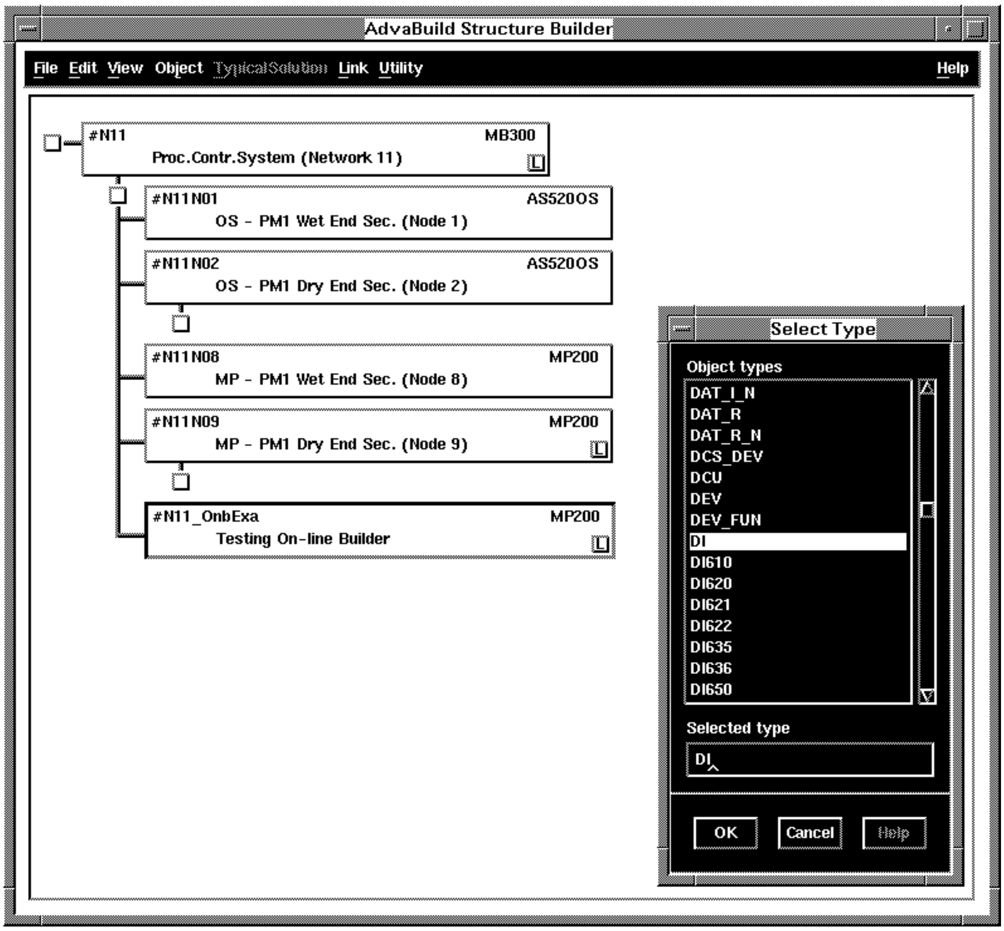
Figure 3-37. Newly inserted list elements

Board type definition for DI and DO board.

NOTE

When you create new boards as described above using STB, the type of the boards will get default values - DSDI 110 for DI boards and DSDO 110 for DO boards. You are not allowed to change the type of the board after creation of it. If you have DI and DO boards in your MasterPiece 200/1 station other than DSDI 110 and DSDO 110, you have to insert these boards using TEB. **Skip the next part, if you use the standard boards:**

- Within the STB, select the MP200 object-type **#N11_OnbExa**.
- Click on **Object** in the menu bar of this window, then click on **Open TYPE...** in the Object menu. You will see the selector box **Select Type**, as in Figure 3-38.

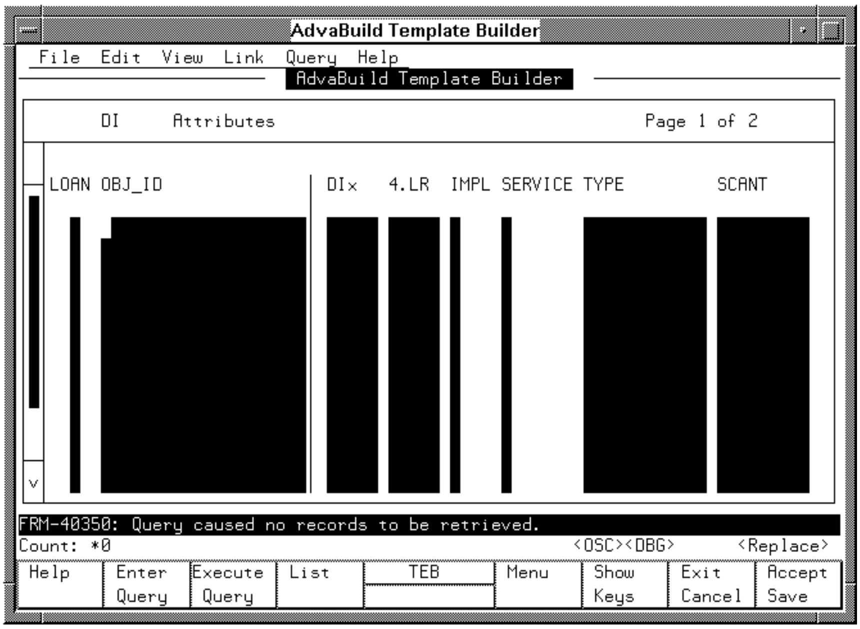

AdvaBuild Structure Builder
File Edit View Object TypicalSolution Link Utility Help
#N11 MB300
Proc.Contr.System (Network 11)
#N11N01 AS520OS
OS - PM1 Wet End Sec. (Node 1)
#N11N02 AS520OS
OS - PM1 Dry End Sec. (Node 2)
#N11N08 MP200
MP - PM1 Wet End Sec. (Node 8)
#N11N09 MP200
MP - PM1 Dry End Sec. (Node 9)
#N11_OnbExa MP200
Testing On-line Builder
Select Type
Object types
DAT_I_N
DAT_R
DAT_R_N
DCS_DEV
DCU
DEV
DEV_FUN
DI
DI610
DI620
DI621
DI622
DI635
DI636
DI650
OK
Cancel
Help

Selected type: DI

Figure 3-38. Selector Box of Open Type and DI is selected

Select **DI** in the Object types and open the TEB in multi element representation:

- Click on the DI object type.
- Press the **OK** button and the TEB will appear see Figure 3-39.

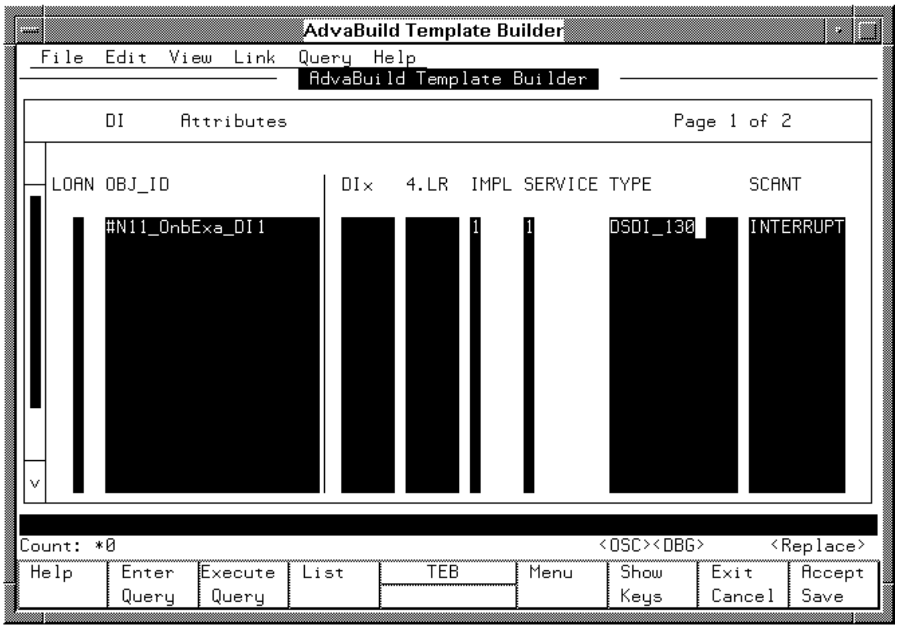


```
FRM-40350: Query caused no records to be retrieved.
Count: *0
```

Figure 3-39. Multi element template for DI

- Click on the **Menu** button.
- Press the **Tab** key, moving the cursor to **Edit**. Press the **Return** key. On **Insert** press the **Return** key again.
- The cursor is in the attribute **OBJ_ID** and you can type in the object id **#N11_OnbExa_DI1** for the new board.
- Press the **Tab** key five times until it is located in the **Type** field.
- Click on the **List** button (in the lower part of the window).
- Select the type you need by moving the cursor with the down arrow key.

– Click on the **Accept Save** button. The new type has been entered (see Figure 3-40).



LOAN	OBJ_ID	DIx	4.LR	IMPL	SERVICE	TYPE	SCANT
	#N11_OnbExa_DI1			1	1	DSDI_130	INTERRUPT

Figure 3-40. Insert the DI board type

– Change other attributes of the board (if needed) in the same way.

– Click on the **Accept Save** button.

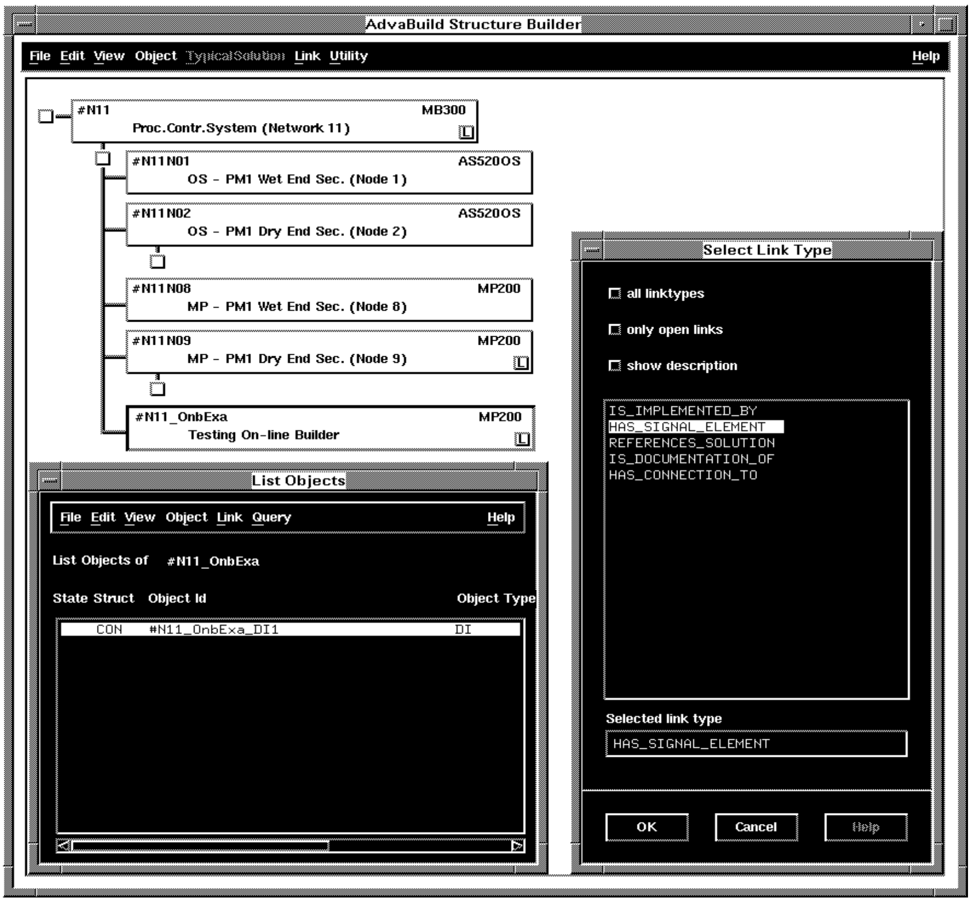
– Leave the Template Builder by clicking on the **Exit Cancel** button.

- If required, do the same procedure for the DO board #N11_OnbExa_DO1.

Link the input and output signals to the DI and DO boards.

The signals (DIS and DOS) have to be linked to the boards before they can be used in the Function Chart Builder.

- In the Structure Builder window, click on the object **#N11_OnbExa**.
- Click on the small rectangle with the label **L** within the same object. You will see a window with a list of DB elements.
- Use the scroll bar on the right hand side of the window to find **#N11_OnbExa_DI1**. Click on this DB element.
- Click on **Link** in the menu bar of this window, then click on **Of Object** in the Link menu. You will see the window **Select Link Type** as in Figure 3-41.


AdvaBuild Structure Builder
File Edit View Object TypicalSolution Link Utility Help
List Objects
File Edit View Object Link Query Help
Select Link Type
OK
Cancel
Help

#N11 MB300
Proc.Contr.System (Network 11)

#N11N01 AS520OS
OS - PM1 Wet End Sec. (Node 1)

#N11N02 AS520OS
OS - PM1 Dry End Sec. (Node 2)

#N11N08 MP200
MP - PM1 Wet End Sec. (Node 8)

#N11N09 MP200
MP - PM1 Dry End Sec. (Node 9)

#N11_OnbExa MP200
Testing On-line Builder

List Objects of #N11_OnbExa

State	Struct	Object Id	Object Type
	CON	#N11_OnbExa_DI1	DI

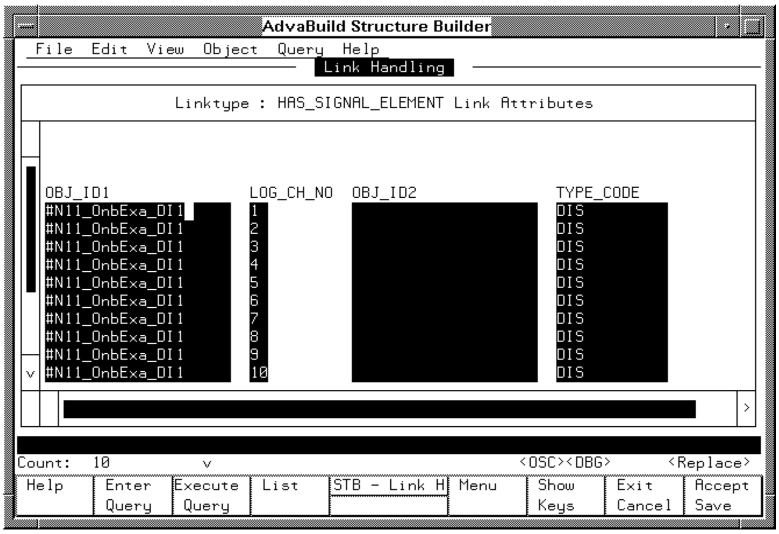
☐ all linktypes
☐ only open links
☐ show description

IS_IMPLEMENTED_BY
HAS_SIGNAL_ELEMENT
REFERENCES_SOLUTION
IS_DOCUMENTATION_OF
HAS_CONNECTION_TO

Selected link type
HAS_SIGNAL_ELEMENT

Figure 3-41. Calling of the Link handling for DI (HAS_SIGNAL_ELEMENT)

- Select the link type **HAS_SIGNAL_ELEMENT** and press the **OK** button. You will see the Link Handling window of the specific Linktype (HSE) as in Figure 3-42 and that all channels of the board are still free.
- Press the **Tab** key twice, moving the cursor to the **OBJ_ID2** field.
  Type **#N11_OnbExa_SWITCH1**.
- Press the down arrow key, moving the cursor to the second line.
  Type **#N11_OnbExa_SWITCH2**.
- Click on the **Accept Save** button.
- Leave the Link Handling window by clicking on the **Exit Cancel** button.
- Repeat the same procedure with DB element #N11_OnbExa_DO1 and the signals #N11_OnbExa_LAMP1 and #N11_OnbExa_LAMP2.
- Leave the list window of DB elements by clicking on **File**/**Close** in the File.



Linktype : HAS_SIGNAL_ELEMENT Link Attributes

OBJ_ID1	LOG_CH_NO	OBJ_ID2	TYPE_CODE
#N11_OnbExa_DI1	1		DIS
#N11_OnbExa_DI1	2		DIS
#N11_OnbExa_DI1	3		DIS
#N11_OnbExa_DI1	4		DIS
#N11_OnbExa_DI1	5		DIS
#N11_OnbExa_DI1	6		DIS
#N11_OnbExa_DI1	7		DIS
#N11_OnbExa_DI1	8		DIS
#N11_OnbExa_DI1	9		DIS
#N11_OnbExa_DI1	10		DIS

Count: 10

Figure 3-42. Link handling window - open links for DI

Create a small PC program

Select the object MP200 named #N11_OnbExa, start-up the Function Chart Builder to initialize the node.
Edit a small program using the existing DI, DO, DIS and DOS and finally generate the Source Code:

- Double click on the object **#N11_OnbExa**.
- In the dialogue box, click on **FUNCT_CHART**, then click on the **OK** button.
  After some seconds, you will see the Function Chart Builder window with an message box, click on the **O**K button.
- Now FCB is started and offer to you a dialog box **ONB.EXA** for creation of a PC program (see Figure 3-43).

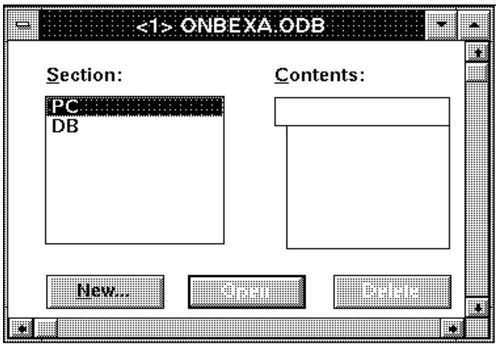
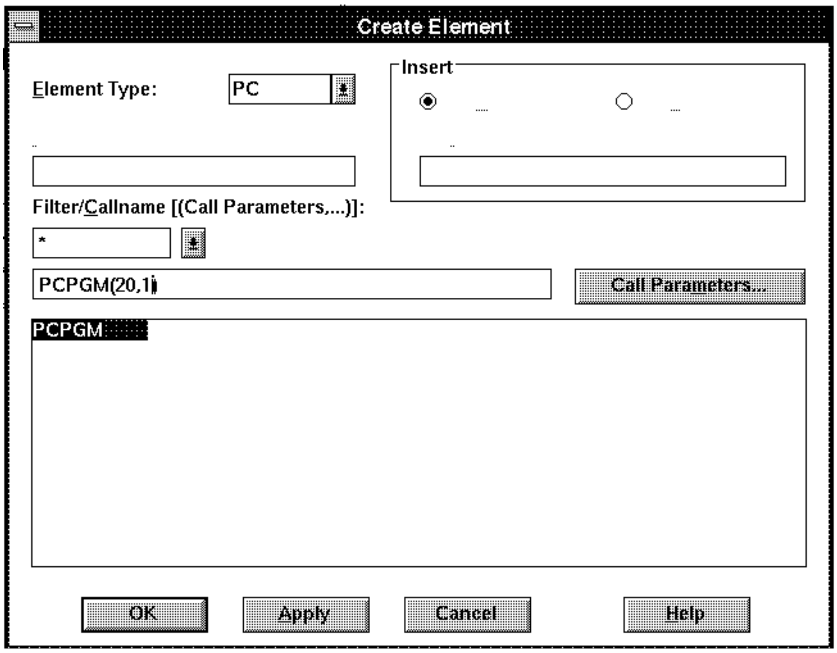


Figure 3-43. Define program name

Select **PC** in Section and click on **New** button.

- Enter **PC1** in field PC Program in upcoming **New PC Program** dialog box and click on **OK** button. Now **Create Element** dialog box is opened (see Figure 3-44).



Element Type: PC

Filter/Callname [(Call Parameters,...)]: *

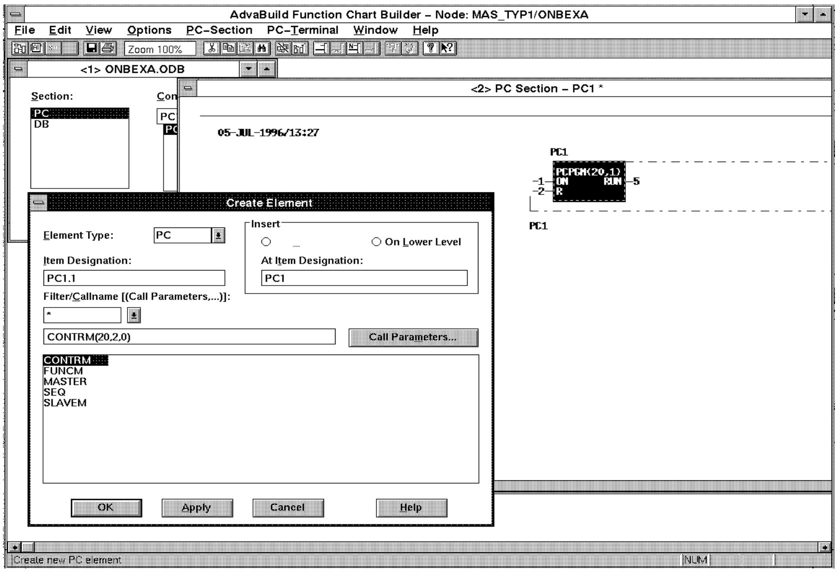
PCPGM(20,1)

PCPGM

Figure 3-44. Create Element

- Click on the **PCPGM** entry and change then call parameter of PCPGM from **(0,0)** to **(20,1)** in edit field.
- Press Apply button for insert.

Now, in the Function Chart Builder window, you can see the function chart as in Figure 3-45. Start editing a small program:



05-JUL-1996/13:27

PCPGM(20,1)

Element Type: PC

Item Designation: PC1.1

At Item Designation: PC1

Filter/Callname [(Call Parameters,...)]: *

CONTRM(20,2,0)

CONTRM
FUNCM
MASTER
SEQ
SLAVEM

Create new PC element

Figure 3-45. Function Chart Builder

Insert a PC element CONTRM

- Click on the rectangle labelled **PCPGM**, click on **PC Element** in the menu bar of the Function Chart Builder window, then click on **Insert On Lower Level** in the PC Element menu.
- In the dialogue box, click on **CONTRM**, then click on the **OK** button.
- In the next dialogue box (**Enter Call Parameter Values**), click on the first field, press the **Backspace** key, then type **20**.
- Click on the second field, press the **Backspace** key, then type **2**.
- Click on the **OK** button.

Insert a PC element AND

- Click on **PC Element** in the menu bar of the Function Chart Builder window, then click on **Insert On Lower Level** in the PC Element menu.
- In the dialogue box, click on **AND**, then click on the **OK** button.
- In the next dialogue box, click on the **OK** button.

Insert a PC element OSC-B

- Click on **PC Element** in the menu bar of the Function Chart Builder window, then click on **Insert On Same Level** in the PC Element menu.
- In the dialogue box, find OSC-B by using the scroll bar. Click on **OSC-B**, then click on the **select** button. In the Function Chart Builder window you will now see all four PC elements (as in Figure 3-46).

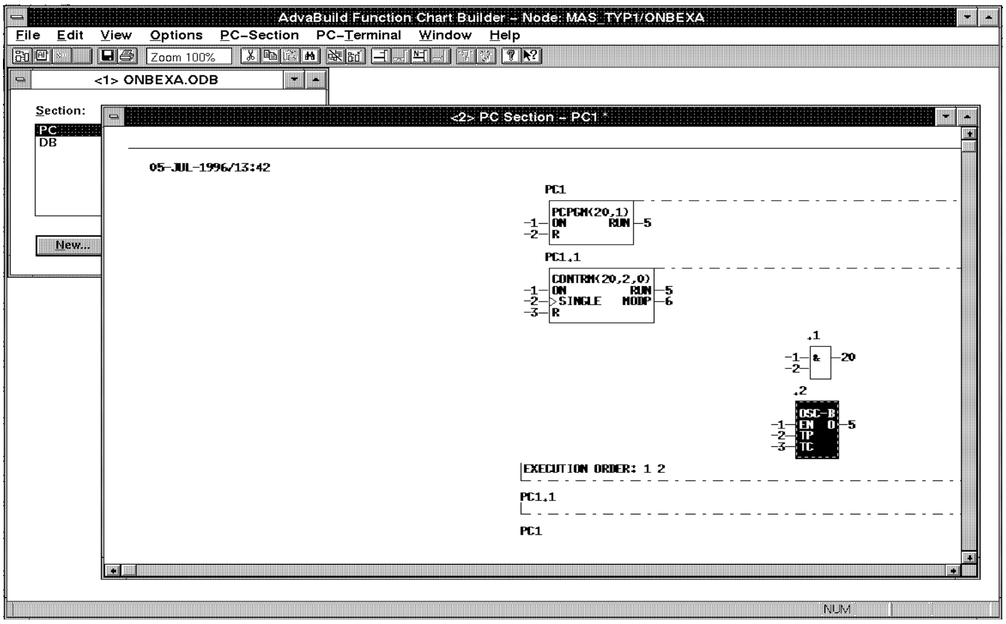


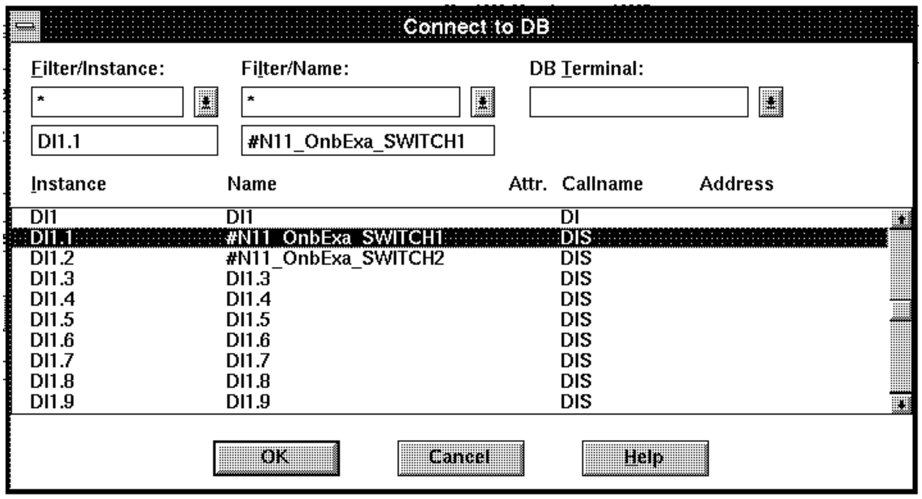
Figure 3-46. FCB with inserted PC elements

Connect PC Terminals to constants

- Click on terminal no. 1 of the PC element PCPGM. Both the terminal and the rectangle will be highlighted.
- Click on **PC Terminal** in the menu bar of the Function Chart Builder window, then click on **Connect...** in the PC Terminal menu.
- In the **Connect** dialogue box field **TO** enter **D=1**, then click on the **Apply/Next** button.
- Enter **D=0** and click on the **Apply/Next** button in **Connect** dialog box, for connecting terminal no. 2 of the PC element PCPGM.
- Click now on button **Next** for toggle to terminal no. 1 of the PC element CONTRM.
- Enter **D=1** in field **TO** then click on the Apply/Next button.
- Repeat the same procedure with terminal no. 2 and terminal no. 3 of the PC element CONTRM, but with value **D=0**.
- Press Clode button for exit of **Connect** dialog box.
- Press now six times for toggle to terminal no. 2 of the PC element OSC-B.
- Enter **D=1s** in field **TO** and click on the **Apply/Next** button for connect of terminal no. 2 of the PC element OSC-B.
- Enter **D=2s** in field **TO** and click on the **Apply/Next** button for connect of terminal no. 3 of the PC element OSC-B.

Connect PC Terminals to DB

- Click on terminal no. 1 of the PC element AND (rectangle labelled **&**).
- Click on button **DB** in the **Connect** dialogue box, now **Connect DB** dialog box is opened see Figure 3-47.



Filter/Instance: *
Filter/Name: *
DB Terminal:

DI1.1
#N11_OnbExa_SWITCH1

Instance	Name	Attr.	Callname	Address
DI1	DI1		DI	
DI1.1	#N11_OnbExa_SWITCH1		DIS	
DI1.2	#N11_OnbExa_SWITCH2		DIS	
DI1.3	DI1.3		DIS	
DI1.4	DI1.4		DIS	
DI1.5	DI1.5		DIS	
DI1.6	DI1.6		DIS	
DI1.7	DI1.7		DIS	
DI1.8	DI1.8		DIS	
DI1.9	DI1.9		DIS	

Figure 3-47. Connect to DB

Click on **#N11_OnbExa_SWITCH1**, then click on the **OK** button.

- In the **Connect** dialogue box, click on the **Apply/Next** button.
- In the same way, connect terminal no. 2 of the PC element **AND** to **#N11_OnbExa_SWITCH2**.
- Click with the **left mouse** button on terminal no. 1 of the PC element OSC-B and then with the right **mouse button** on terminal no. 2 of the PC element AND, for connecting.
- Click on terminal no. 20 of the PC element AND (rectangle labelled **&**).
- Click on button **DB** in the **Connect** dialogue box, now **Connect DB** dialog box is opened again.
- Scroll with slide in the **Connect DB** dialogue box to entry **#N11_OnbExa_LAMP1.**
- Click on **#N11_OnbExa_LAMP1**, then click on the **OK** button.
- In the same way, connect terminal no. 5 of the PC element **OSC-B** to **#N11_OnbExa_LAMP2**.

Enlarge the Function Chart Builder PC-Section window to see the whole picture as in Figure 3-48.

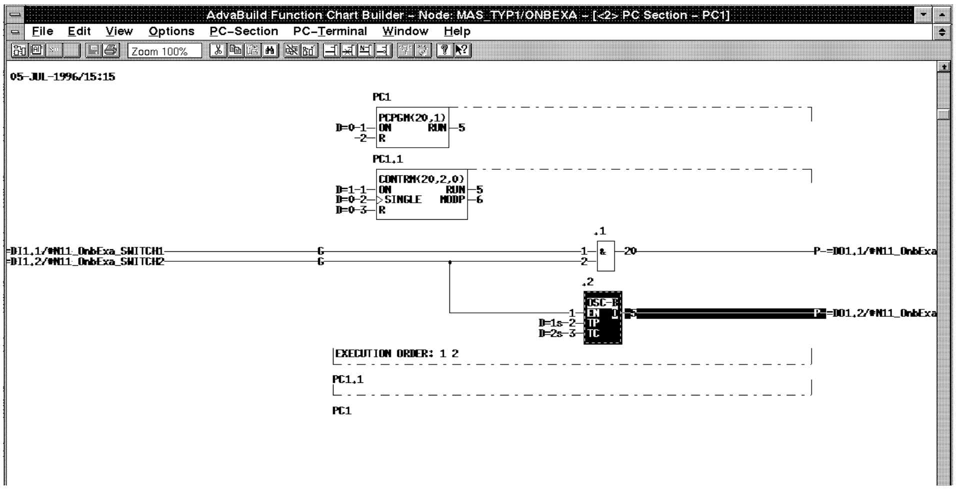


Figure 3-48. Small PC program

Make some notes about your input and output signals.

- Take a sheet of paper and write down the board and channel numbers of #N11_OnbExa_SWITCH1, #N11_OnbExa_SWITCH2, #N11_OnbExa_LAMP1 and #N11_OnbExa_LAMP2 as documented in the function chart.
  In Figure 3-48they are indicated as DI1.1 , DI1.2 , DO1.1 and DO1.2.

Generate Source Code

- Click on **File-Generate Source** in the menu bar of the Function Chart Builder window.
- Mark in new **Generate Source Code** dialog box **PC Section** and **DB Section**.
- Enter **PC** for **PC File Name** and **DB** for **DB File Name** and press **OK** button.

Leave the Function Chart Builder

- Click on **File Exit** in the menu bar of the Function Chart Builder window, then click on **Exit** in the System menu.

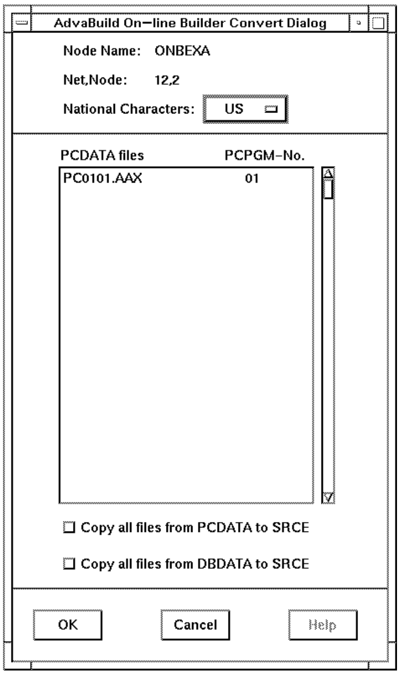
Now all preparations have been done, you are ready to start the On-line Builder.

3.3.3 Using the On-line Builder

Starting the On-line Builder

Start the On-line Builder using the MP200 object named #N11_OnbExa prepared in the previous section.

- In the Structure Builder window, double click on the object **#N11_OnbExa**.
- In the dialogue box, click on **ON-LINE**, then click on the **OK** button. You will see a dialogue box as in Figure 3-49.



AdvaBuild On–line Builder Convert Dialog

Node Name: ONBEXA

Net,Node: 12,2

National Characters: US

PCDATA files	PCPGM–No.
PC0101.AAX	01

☐ Copy all files from PCDATA to SRCE

☐ Copy all files from DBDATA to SRCE

Figure 3-49. On-line Builder start-up

Copy Source Code Files for Down-load

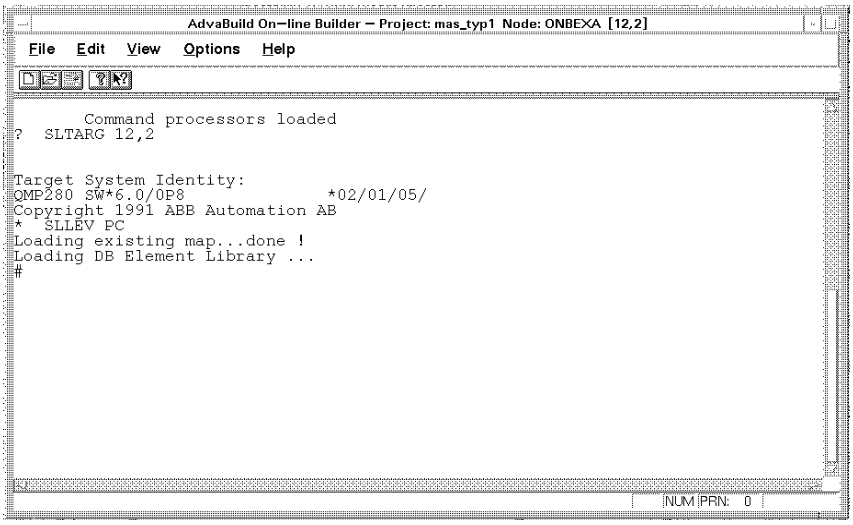
Right now, you will find the source code files in the directories of DBDATA and of PCDATA.

These files may contain special national characters like Å and Ü. The files are coded in the 8bit ISO88591 code. For downloading, the files must be in the directory of SRCE, and they must be coded in the 7bit ASCII code. The On-line Builder will do these convert and copy functions.

- In the dialogue box, click on the small rectangle left of the text **Copy all files from PCDATA to SRCE**.
- Click on the small rectangle left of the text **Copy all files from DBDATA to SRCE**.
- Click on the **OK** button.

Establish Connection to Target Node

The On-line Builder will convert and copy the files to the SRCE directory, load the command processor software for MasterPiece 200/1 (as defined by the type and attributes of the object #N11_OnbExa) into the real-time accelerator board and then request the real-time accelerator board to establish a connection to the target node (network and node address as defined by the attributes of the object #N11_OnbExa). After some seconds, you will see the On-line Builder window, as in Figure 3-50.


AdvaBuild On–line Builder – Project: mas_typ1 Node: ONBEXA [12,2]
File
Edit
View
Options
Help
NUM
PRN: 0

```
       Command processors loaded
?  SLTARG 12,2


Target System Identity:
QMP280 SW*6.0/0P8              *02/01/05/
Copyright 1991 ABB Automation AB
*  SLLEV PC
Loading existing map...done !
Loading DB Element Library ...
#
```

Figure 3-50. On-line Builder window

The network and node number of the target node is shown on the upper border of the On-line Builder window. The * prompt tells you the command processor is on the system level.

Show the system identification of the target node.

- Type **SHTARG <Return>**

Show the status of the application program on the target node.

- Type **LSS <Return>**

CAUTION
Be aware the target node isn't connected to a hot process configuration, because the next steps will change the status of output signals (and programs and definitions of I/O channels, too). If you are not quite sure, leave the session right now.

Down-load Program to Target Node

You will switch the target node to configuration mode, do the database and PC area dimensioning and then down-load the program to the target node.

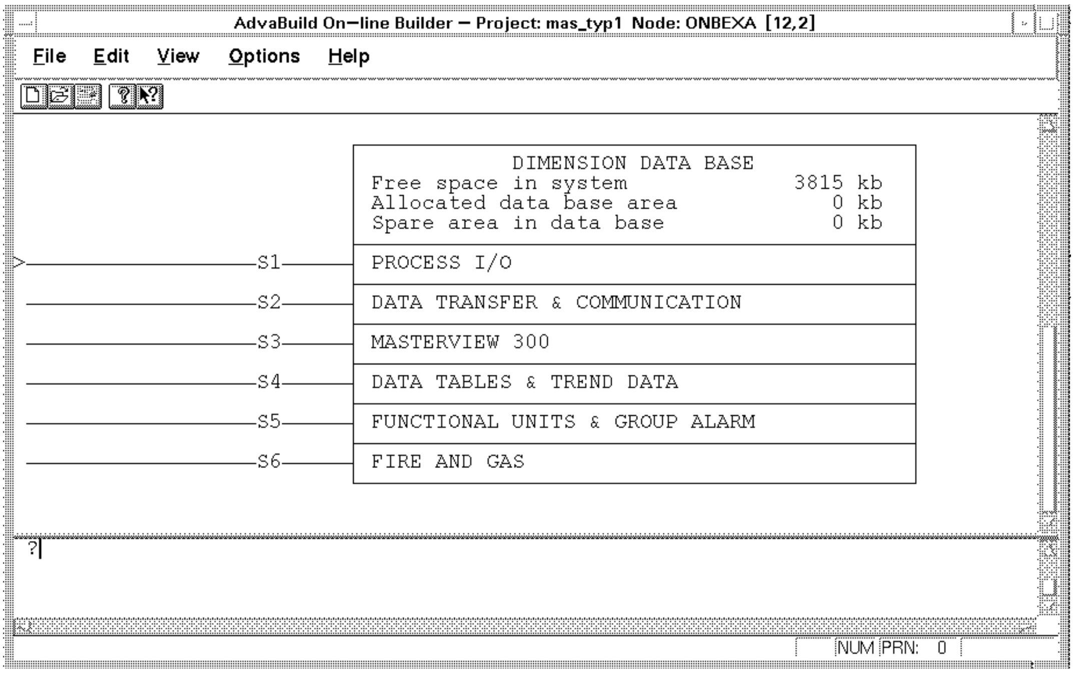
Switch target node to Configuration mode

- Type **RECONFIG <Return>**
- Answer the question by typing **Y <Return>**.

Dimension the database of the target node

- Type **DIMDB <Return>**

Now the window will be spitted in a graphical and command section, see Figure 3-51.

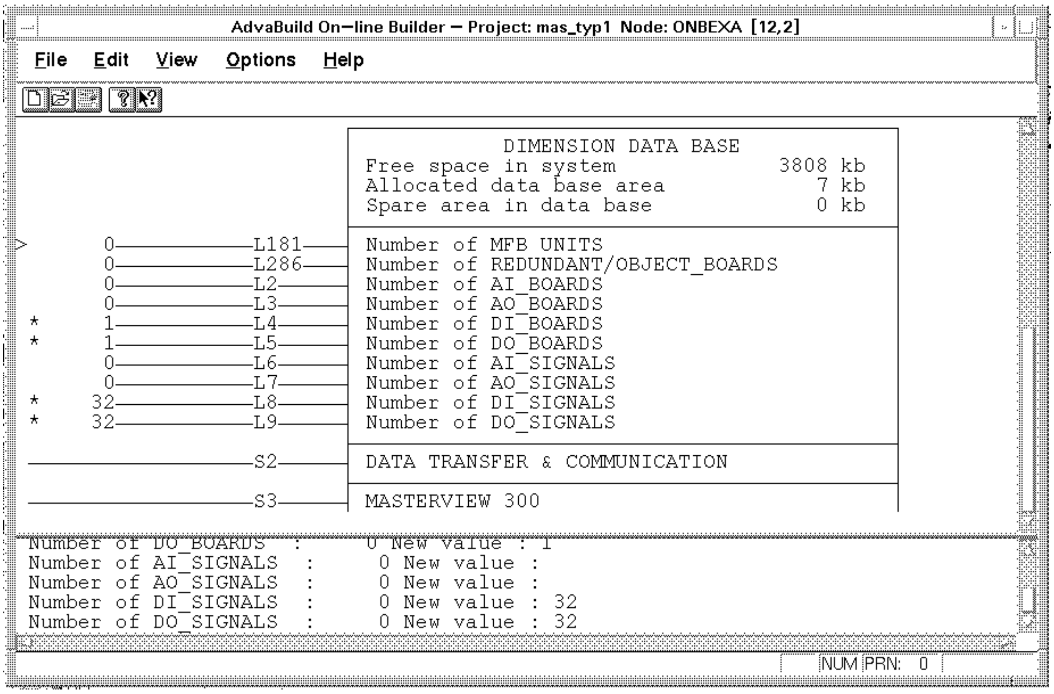

AdvaBuild On–line Builder – Project: mas_typ1 Node: ONBEXA [12,2]
File
Edit
View
Options
Help
S1
S2
S3
S4
S5
S6
?
NUM
PRN: 0

```
             DIMENSION DATA BASE
Free space in system             3815 kb
Allocated data base area            0 kb
Spare area in data base             0 kb
PROCESS I/O
DATA TRANSFER & COMMUNICATION
MASTERVIEW 300
DATA TABLES & TREND DATA
FUNCTIONAL UNITS & GROUP ALARM
FIRE AND GAS
```

Figure 3-51. On-line Builder window during DIMDB

- Move the mouse pointer into the command section of **AdvaBuild On-line Builder** window.
- Type **S1<Return>**. The picture in the graphical section will change.
- Press the **Return** key four times, thus moving the cursor in the graphic section to the line which reads **Number of DI_BOARDS**.
- Type **M <Return>**, then type **1 <Return>**.
- Type **1 <Return>**.
- Press the **Return** key two times, thus moving the cursor in the graphic section to the line which reads **Number of DI_SIGNALS**.
- Type **32 <Return>**.

- Type **32 <Return>**. Your screen should look like Figure 3-52.


AdvaBuild On–line Builder – Project: mas_typ1 Node: ONBEXA [12,2]
File
Edit
View
Options
Help
NUM
PRN: 0

```
                                  DIMENSION DATA BASE
                          Free space in system            3808 kb
                          Allocated data base area           7 kb
                          Spare area in data base            0 kb

>      0————————————L181————  Number of MFB UNITS
       0————————————L286————  Number of REDUNDANT/OBJECT_BOARDS
       0————————————L2——————  Number of AI_BOARDS
       0————————————L3——————  Number of AO_BOARDS
 *     1————————————L4——————  Number of DI_BOARDS
 *     1————————————L5——————  Number of DO_BOARDS
       0————————————L6——————  Number of AI_SIGNALS
       0————————————L7——————  Number of AO_SIGNALS
 *    32————————————L8——————  Number of DI_SIGNALS
 *    32————————————L9——————  Number of DO_SIGNALS

 ———————————————————S2——————  DATA TRANSFER & COMMUNICATION

 ———————————————————S3——————  MASTERVIEW 300
```

```
Number of DO_BOARDS   :     0 New value : 1
Number of AI_SIGNALS  :     0 New value :
Number of AO_SIGNALS  :     0 New value :
Number of DI_SIGNALS  :     0 New value : 32
Number of DO_SIGNALS  :     0 New value : 32
```

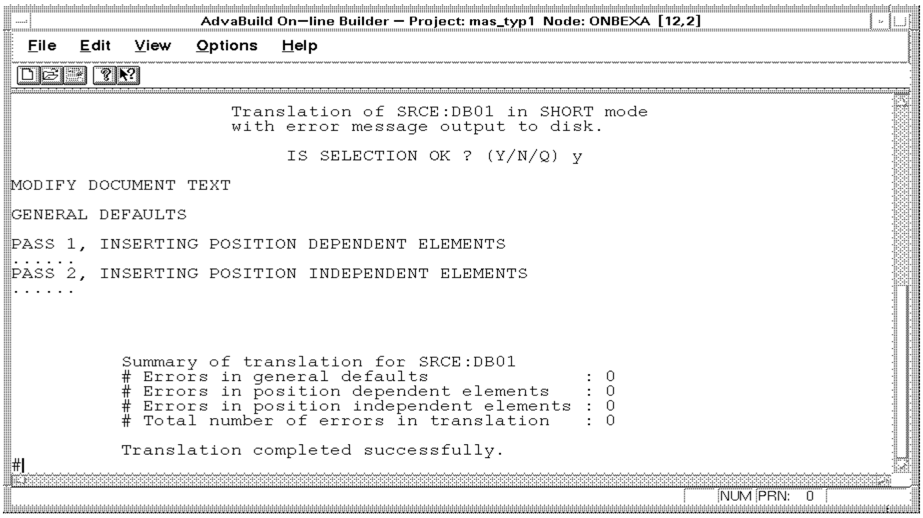
Figure 3-52. On-line Builder window during DIMDB

- Type **! <Return>**, then type **DIM <Return>**.
- Type **100 <Return>**, specifying the spare area.

Down-load DB part of the application program to the target node

- Type **TRDBS <Return>**.
- Answer the question by typing **DB01 <Return>**.
- Answer the next question by typing **S <Return>**.
- Answer the next question by typing **Y <Return>**.

It will take about a minute to down-load and translate this file. then you will get a message as in Figure 3-53.

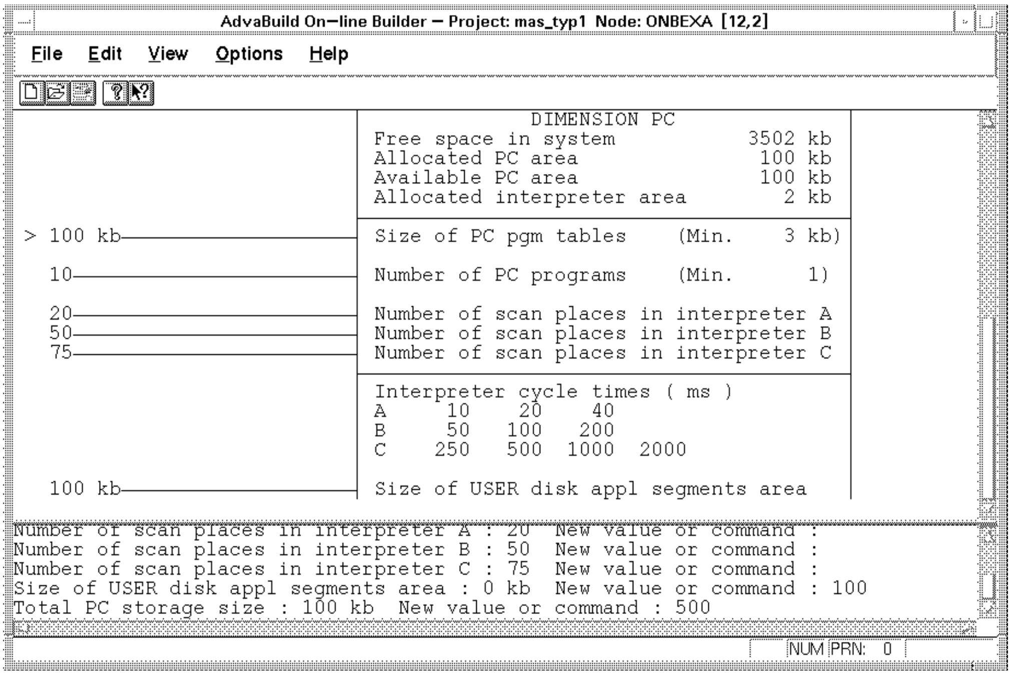

AdvaBuild On—line Builder — Project: mas_typ1 Node: ONBEXA [12,2]
File
Edit
View
Options
Help
NUM
PRN: 0

```
                    Translation of SRCE:DB01 in SHORT mode
                    with error message output to disk.

                         IS SELECTION OK ? (Y/N/Q) y

MODIFY DOCUMENT TEXT

GENERAL DEFAULTS

PASS 1, INSERTING POSITION DEPENDENT ELEMENTS
......
PASS 2, INSERTING POSITION INDEPENDENT ELEMENTS
......



          Summary of translation for SRCE:DB01
          # Errors in general defaults              : 0
          # Errors in position dependent elements   : 0
          # Errors in position independent elements : 0
          # Total number of errors in translation   : 0

          Translation completed successfully.
#|
```

Figure 3-53. On-line Builder window after TRDBS (Down-load)

Dimension the PC area of the target node

- Type **DIMPC <Return>**.
- Type **M <Return>**.
- Press the **Return** key five times, thus moving the cursor in the graphic section to the line which reads **Size of USER disk appl. segments area**.
- Type **100 <Return>**.
- Type **500 <Return>**, specifying the total PC storage size.
  Your screen will look like Figure 3-54.
- Type **! <Return>**, then type **DIM <Return>**.



AdvaBuild On—line Builder — Project: mas_typ1 Node: ONBEXA [12,2]
File
Edit
View
Options
Help
NUM
PRN: 0

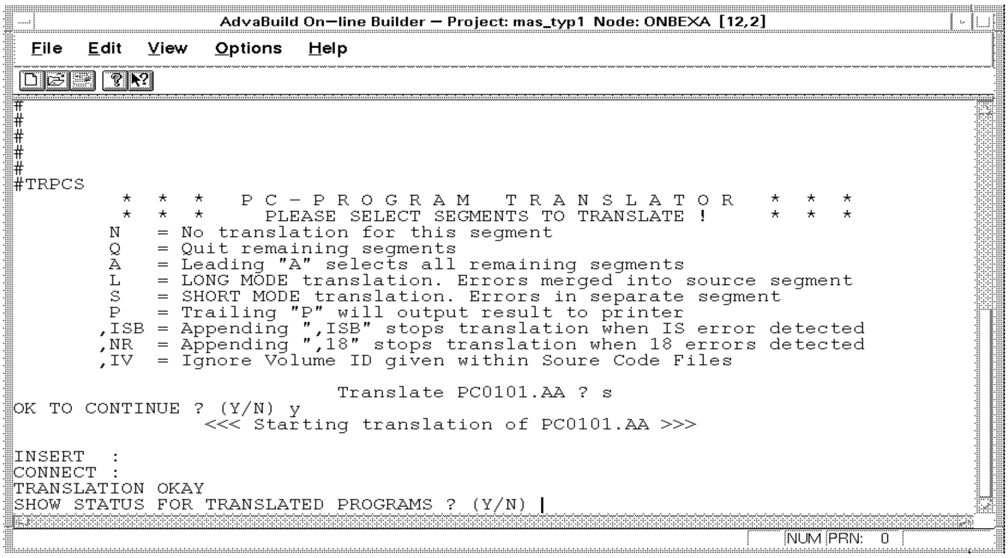
```
                                   DIMENSION PC
                           Free space in system           3502 kb
                           Allocated PC area               100 kb
                           Available PC area               100 kb
                           Allocated interpreter area        2 kb

 > 100 kb                  Size of PC pgm tables    (Min.    3 kb)

   10                      Number of PC programs    (Min.       1)

   20                      Number of scan places in interpreter A
   50                      Number of scan places in interpreter B
   75                      Number of scan places in interpreter C

                           Interpreter cycle times ( ms )
                           A     10    20    40
                           B     50   100   200
                           C    250   500  1000  2000

   100 kb                  Size of USER disk appl segments area
```

```
Number of scan places in interpreter A : 20  New value or command :
Number of scan places in interpreter B : 50  New value or command :
Number of scan places in interpreter C : 75  New value or command :
Size of USER disk appl segments area : 0 kb  New value or command : 100
Total PC storage size : 100 kb  New value or command : 500
```

Figure 3-54. On-line Builder window during DIMPC

Down-load PC part of the application program to the target node.

- Type **TRPCS <Return>**.
- Answer the question by typing **S <Return>**.
- Answer the next question by typing **Y <Return>**. It will take some seconds to down-load and translate this file. Then you will get a message as in Figure 3-55.



AdvaBuild On—line Builder — Project: mas_typ1 Node: ONBEXA [12,2]
File
Edit
View
Options
Help
NUM
PRN: 0

```
#
#
#
#
#
#TRPCS
         *  *  *   P C - P R O G R A M   T R A N S L A T O R   *  *  *
         *  *  *     PLEASE SELECT SEGMENTS TO TRANSLATE !     *  *  *
        N   = No translation for this segment
        Q   = Quit remaining segments
        A   = Leading "A" selects all remaining segments
        L   = LONG MODE translation. Errors merged into source segment
        S   = SHORT MODE translation. Errors in separate segment
        P   = Trailing "P" will output result to printer
       ,ISB = Appending ",ISB" stops translation when IS error detected
       ,NR  = Appending ",18" stops translation when 18 errors detected
       ,IV  = Ignore Volume ID given within Soure Code Files

                           Translate PC0101.AA ? s
OK TO CONTINUE ? (Y/N) y
                <<< Starting translation of PC0101.AA >>>

INSERT  :
CONNECT :
TRANSLATION OKAY
SHOW STATUS FOR TRANSLATED PROGRAMS ? (Y/N) |
```

Figure 3-55. On-line Builder window after TRPCS (Down-load)

- Type **Y <Return>** to see further informations about the translation.
- Press the **Return** key.

Test and modify the Application Program

You will switch the target node to Operation mode and see actual signals of the running target. Then you'll do on-line modifications to the PC program.

Show the status of the application program on the target node.

- Type **LSS <Return>**. You will get a message, telling you the system is in Configuration mode and PC1 is in Build mode.

Switch target node to Operation mode.

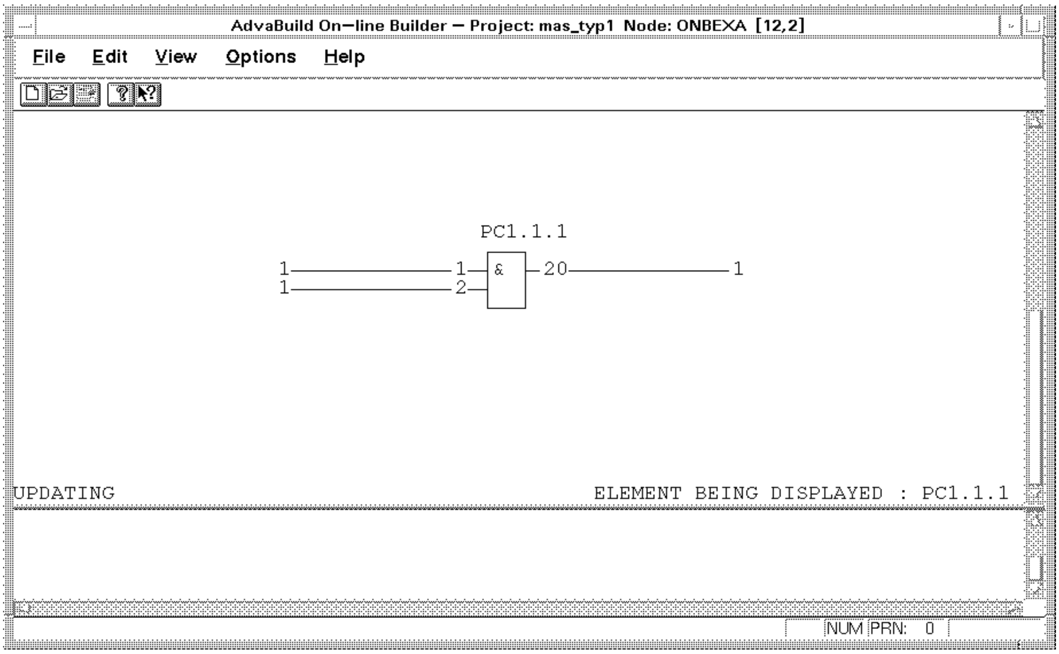
- Type **DICONFIG <Return>**.
- Answer the question by typing **Y <Return>**.
  After some seconds you will get a message telling you the system is in Operation mode.

Switch program PC1 to Unblocked mode

- Type **DIBM PC1 <Return>**, then type **DBL PC1 <Return>**.

Show the PC element AND (which has the item number PC1.1.1) with dynamic updating of its input and output values.

- Type **GEPCD D, PC1.1.1 <Return>**.
  On the On-line Builder window you will see the PC element AND as in Figure 3-56.


AdvaBuild On–line Builder – Project: mas_typ1 Node: ONBEXA [12,2]
File
Edit
View
Options
Help
PC1.1.1
1
1
1
2
&
20
1
NUM
PRN: 0

```
UPDATING                                ELEMENT BEING DISPLAYED : PC1.1.1
```

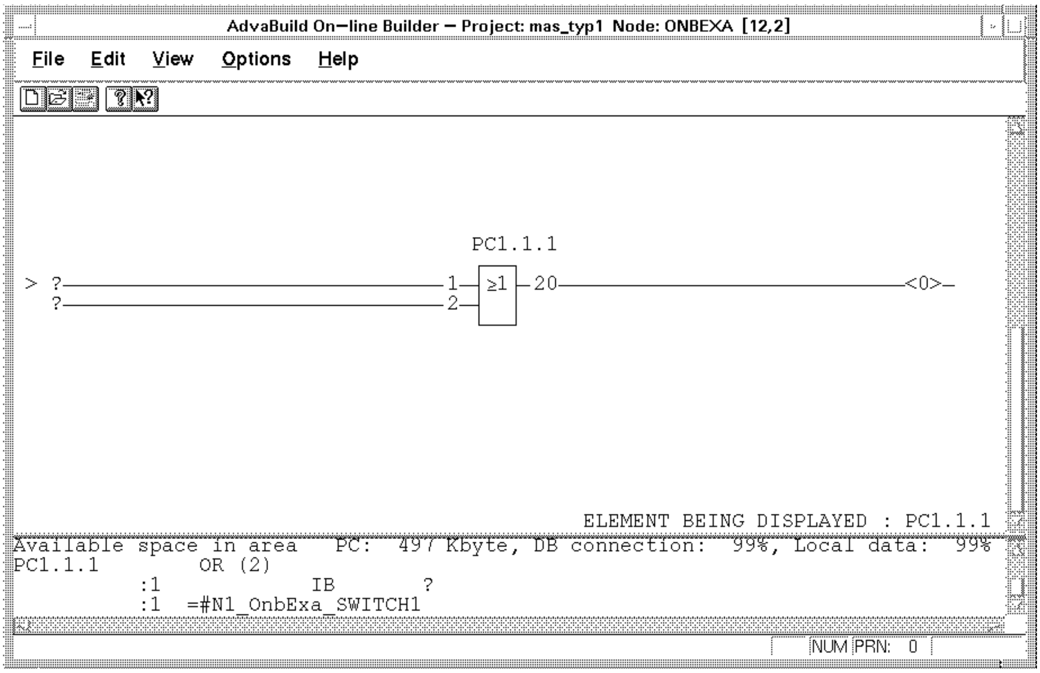
Figure 3-56. On-line Builder window 'AND' with dynamic updating

- Type **<CTRL-B>** to stop the dynamic updating.
- Type **! <Return>.**

Have a look on the hardware of the target node. See whether it's running properly by switching SWITCH1 and SWITCH2 and watching LAMP1 and LAMP2. Consult your paper about the assignments of those signals to the channels (DI1.x, DO1.y) of the boards DI1 and DO1. Repeat the dynamic updating as above, if you like.

Delete the PC element AND and instead insert a PC element OR.

- First, switch program PC1 to Build mode by typing **EBM PC1 <Return>**.
- Type **DS PC1.1.1 <Return>** to delete the PC element AND.
- Answer the question by typing **Y <Return>**.
- Answer the next question by typing **N <Return>**.
- Type **IS PC1.1 <Return>**, type **OR (2) <Return>**, then type **! <Return>** in order to insert a PC element OR with two inputs.
- Type **C PC1.1.1:1 <Return>**. In the On-line Builder window, you will see the PC element OR as in Figure 3-57.


AdvaBuild On–line Builder – Project: mas_typ1 Node: ONBEXA [12,2]
File
Edit
View
Options
Help
PC1.1.1
> ?
?
1
2
≥1
20
<0>
NUM
PRN: 0

```
ELEMENT BEING DISPLAYED : PC1.1.1
Available space in area   PC:  497 Kbyte, DB connection:  99%, Local data:  99%
PC1.1.1        OR (2)
         :1           IB       ?
         :1  =#N1_OnbExa_SWITCH1
```

Figure 3-57. On-line Builder window 'OR' while connecting terminals

- Type **=#N11_OnbExa_SWITCH1 <Return>**, type **=#N11_OnbExa_SWITCH2 <Return>**, type **=#N11_OnbExa_LAMP1 <Return>**, then type **! <Return>**, thus connecting the PC element OR to the inputs and to the output.

Switch program PC1 to Unblocked mode.

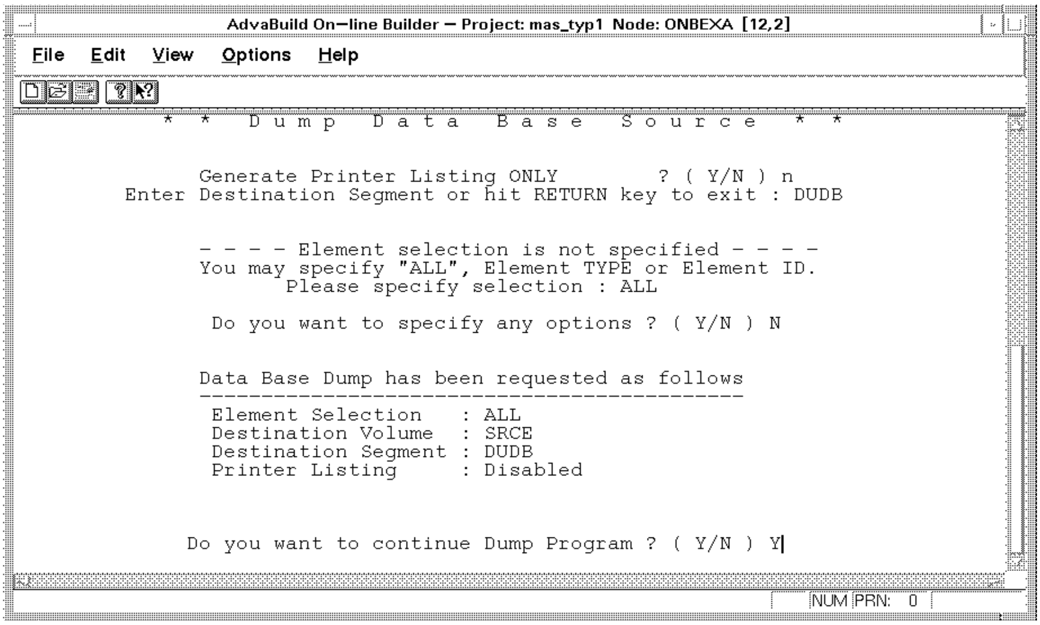
- Type **DIBM PC1 <Return>**, then type **DBL PC1 <Return>**.

If you like, you may test the modified program as described above.

Up-load Program from Target Node

Up-load DB part of the application program from the target node.

- Type **DUDBS <Return>**
- Answer the question by typing **N <Return>**.
- Type **DUDB <Return>**, then type **ALL <Return>**.
- Answer the question by typing **N <Return>**. The On-line Builder window will look like Figure 3-58.


AdvaBuild On—line Builder — Project: mas_typ1 Node: ONBEXA [12,2]
File
Edit
View
Options
Help
NUM
PRN: 0

```
          * *  D u m p   D a t a   B a s e   S o u r c e   * *


               Generate Printer Listing ONLY        ? ( Y/N ) n
         Enter Destination Segment or hit RETURN key to exit : DUDB


               - - - - Element selection is not specified - - - -
               You may specify "ALL", Element TYPE or Element ID.
                      Please specify selection : ALL

                Do you want to specify any options ? ( Y/N ) N


               Data Base Dump has been requested as follows
               --------------------------------------------
                Element Selection    : ALL
                Destination Volume   : SRCE
                Destination Segment  : DUDB
                Printer Listing      : Disabled



              Do you want to continue Dump Program ? ( Y/N ) Y
```

Figure 3-58. On-line Builder window during DUDBS (Up-load)

- Answer the next question by typing **Y <Return>**. After a minute, you will get a message telling you up-load (dump) has been done.

Up-load PC part of the application program from the target node.

- Type **DUPCS PC1/SRCE:DUPC <Return>**.
- Answer the question by typing **Y <Return>**.

Disconnect from Target Node

- Type **<CTRL-C>**.
  The On-line Builder window disappear.

Copy Source Code Files for Backtranslate

The files that have been up-loaded from the target node are now in the directory of SRCE. To be used in the Function Chart Builder, they have to be converted and copied to the directories of DBDATA and PCDATA. The On-line Builder will do this convert and copy functions.

- Click on the **Yes** button in the dialogue box.

Leave the On-line Builder

- Click on the **Yes** button in the next dialogue box. After some seconds, the On-line Builder dialogue box disappears.

Integrate modified Application Program into Project Structure

The old MP200 PC program has to be deleted before you can backtranslate the application program. You will backtranslate from Source Code (as delivered by the On-line Builder) to the form used by the Function Chart Builder. Finally you'll see the program up-loaded from target node in Function Chart representation.

Call the Function Chart Builder.

- Double click on the MP200 object **#N11_OnbExa**.
- In the dialogue box, click on **FUNCT_CHART**, then click on the **OK** button.
  After some seconds, you will see the Function Chart Builder window with an message box, click on the **O**K button.

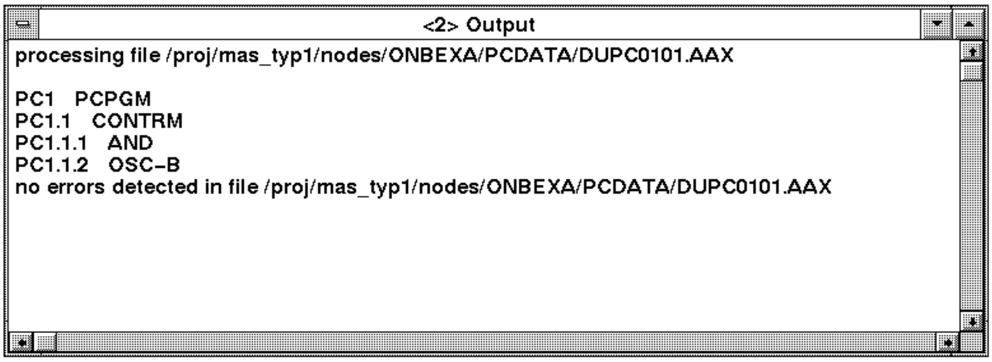
Delete program PC1

When Backtranslating, the Function Chart Builder will not overwrite an existing PC program. Therefore you have to delete it first.

- Click on **File - Delete Section** and there **PC** submenu in the menu bar of the Function Chart Builder window.
- In the dialogue box, click on **PC1**, then click on the **OK** button.
- Answer the question by clicking on the **YES** button.
- Answer the question by clicking on the **OK** button.

Backtranslate the application program

- Click on **File - Backtranslate Source** in the menu bar of the Function Chart Builder window.
- Click on **PC Section** and click off **DB Section** in the dialog box.
- Select program **DUPC** in **PC Section** and click **OK** button.
- In the new dialogue box click on the **YES** button.
  After some seconds, the message window will look like Figure 3-59.



```
processing file /proj/mas_typ1/nodes/ONBEXA/PCDATA/DUPC0101.AAX

PC1   PCPGM
PC1.1   CONTRM
PC1.1.1   AND
PC1.1.2   OSC-B
no errors detected in file /proj/mas_typ1/nodes/ONBEXA/PCDATA/DUPC0101.AAX
```

Figure 3-59. Backtranslate complete

Have a look on the modified application program.

- Click in **ONBEXA.ODB** window on **PC1** and press **Open** button.
  In the Function Chart Builder window, you will see the four PC elements of the modified program, as in Figure 3-60.

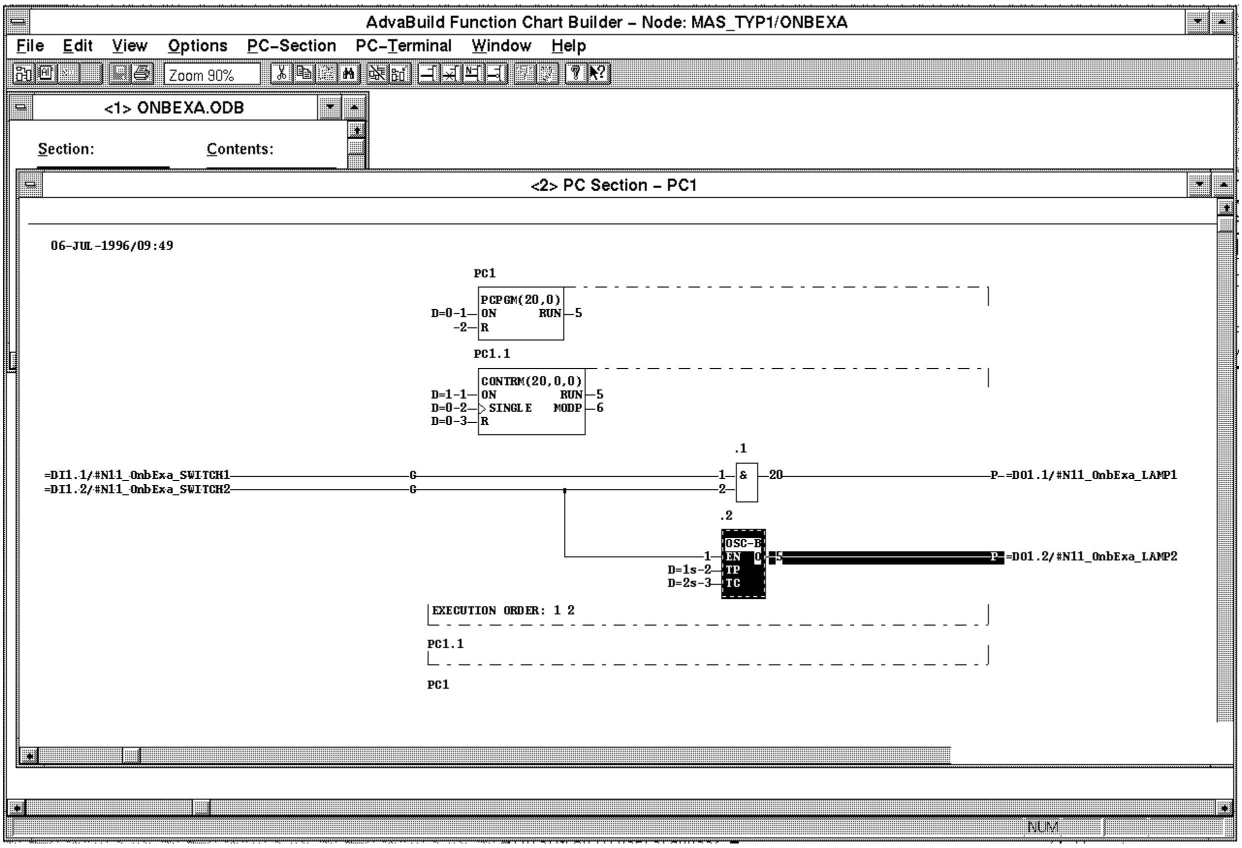


Figure 3-60. Modified PC program

- Be happy, your exercise has been successful.

Leave the Function Chart Builder

- Click on **File** in the menu bar of the Function Chart Builder window, then click on **Exit** in the System menu.

Delete the object #N11_OnbExa as explained at the beginning of section 4.3.2, if you don't need it any longer.

3.4 Application Procedures

This section describes the following basic application procedures:

- How to use the Administration function to create new projects and assign project users
- How to build the project structure (insert objects, copy objects, and so on)
- How to edit object attributes
- How to link objects in different structures
- How to configure the Plant Documentation Builder and print out documentation packages
- How to generate custom database reports
- How to load text and drawing files from a floppy disk to the Text and Diagram Builders
- How to import ASCII files from diskette to the Advant Station
- How to enter and execute database queries.

This section provides guidelines for basic application procedures. If you need further information on a specific menu item for one of these procedures, refer to Section 3.5, Menus.

3.4.1 Application Procedures For Administration

3.4.1.1 Administration Functions Overview

This section provides some background information regarding the AdvaBuild administration functions. Instructions on how to use these functions are provided in the sections that follow. The Administration functions for projects are can be used to:

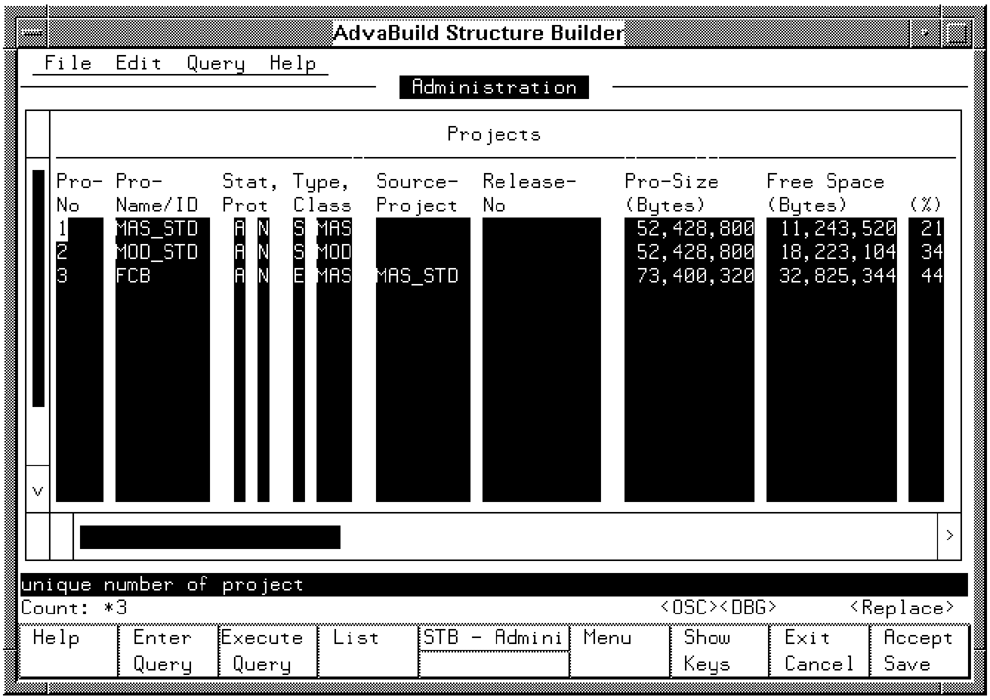
- insert (create) a new project.
- delete a project.
- save a project as a project release.
- restore a project saved as a project release.
- enlarge project size.
- edit comments and change the protection status.
- print out project information.

Administration functions for Users can be used to:

- add (insert) users to a project.
- delete users from the system.
- change user passwords.

Administration Windows, Dialog Boxes and Information Boxes

The primary windows for AdvaBuild Administration functions are the Projects Window, Figure 3-43, and Users Window, Figure 3-44. These windows show overall information for projects and users, they are not for editing data. To insert new projects or users, use the menu item **Edit Insert** in the respective window.



Projects

Pro-No	Pro-Name/ID	Stat, Prot		Type, Class		Source-Project	Release-No	Pro-Size (Bytes)	Free Space (Bytes)	(%)
1	MAS_STD	A	N	S	MAS			52,428,800	11,243,520	21
2	MOD_STD	A	N	S	MOD			52,428,800	18,223,104	34
3	FCB	A	N	E	MAS	MAS_STD		73,400,320	32,825,344	44

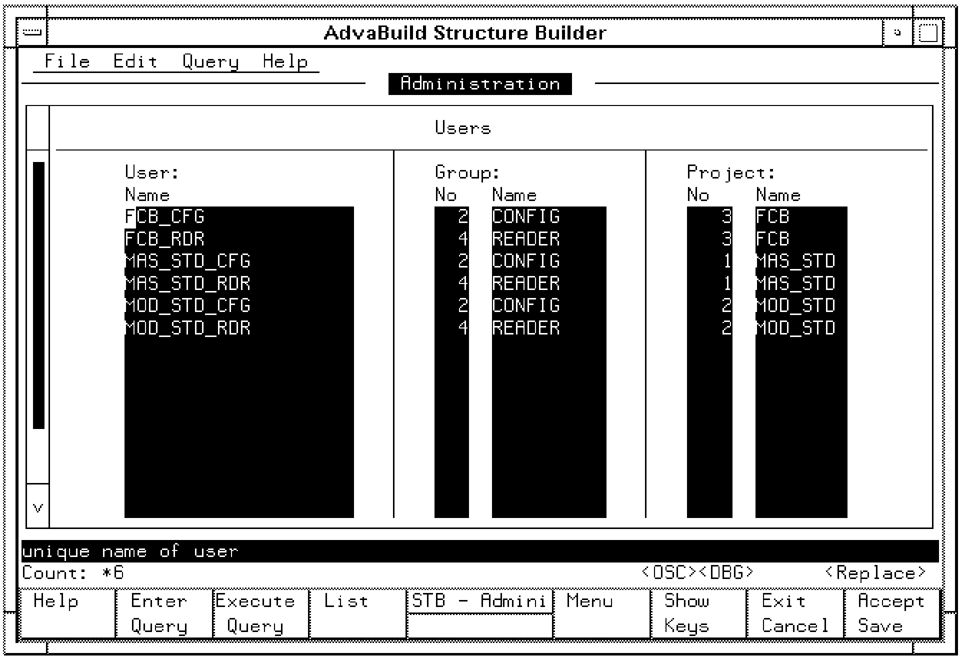
unique number of project
Count: *3 <OSC><DBG> <Replace>

Figure 3-43. Projects Window

The Projects window provides a list of existing projects, and indicates the project status, protection, type and class as well as the total and the free project space, the source project and any restored project releases. For more detailed information refer to Section 3.5.3, Menus for Projects Window.

The menu bar provides functions for inserting new projects, deleting old projects, enlarging a project's size, saving and restoring project releases and modifying a project's comment and protection mode as well as for printing out project information as shown in the Projects window.

The Users window, Figure 3-44, shows a list of users, and indicates the user group and the projects they belong to.



```
                              Users
User:                 Group:                Project:
Name                  No   Name             No   Name
FCB_CFG                2   CONFIG            3   FCB
FCB_RDR                4   READER            3   FCB
MAS_STD_CFG            2   CONFIG            1   MAS_STD
MAS_STD_RDR            4   READER            1   MAS_STD
MOD_STD_CFG            2   CONFIG            2   MOD_STD
MOD_STD_RDR            4   READER            2   MOD_STD

unique name of user
Count: *6                                <OSC><DBG>      <Replace>
```

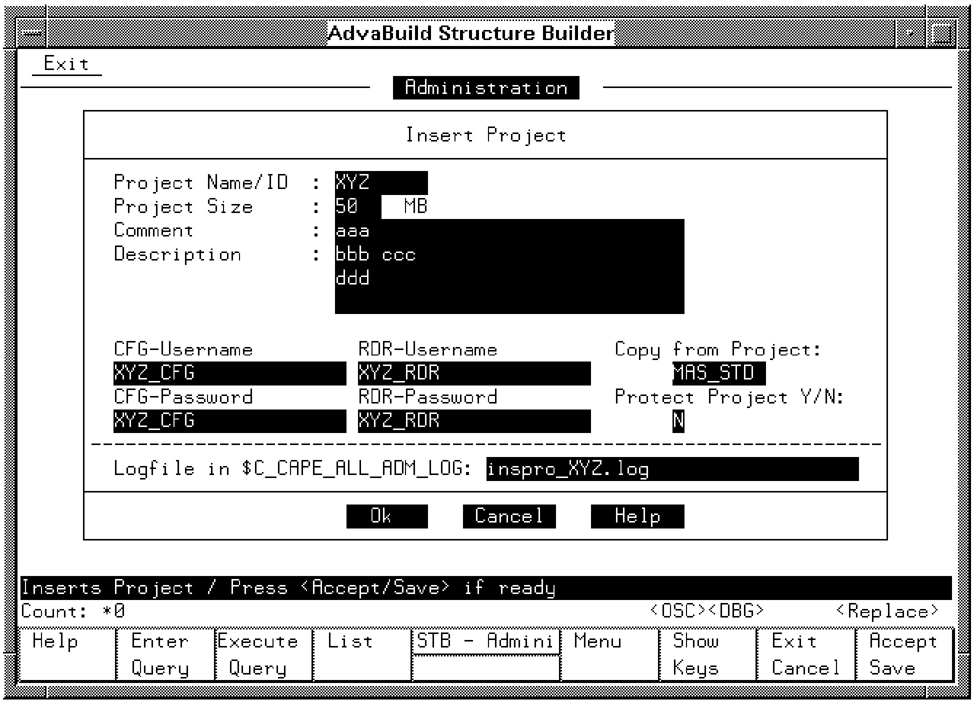
Figure 3-44. Users Window

The menu bar of the Users window provides functions for inserting new users, deleting old ones and changing a user's password as well as for printing out user information as shown in the Users window.

NOTE

> Do not press <CTRL-S> or <CTRL-Q> while in any of the Administration windows and boxes; otherwise, cursor movements or terminal output will not be visible anymore.

After you start an Administration function by choosing a menu item in the Projects or Users window, the applicable dialog box is displayed. An example dialog box for menu item **Projects Edit Insert** is shown in Figure 3-45.



Project Name/ID : XYZ
Project Size : 50 MB
Comment : aaa
Description : bbb ccc
ddd

CFG-Username XYZ_CFG
RDR-Username XYZ_RDR
Copy from Project: MAS_STD
CFG-Password XYZ_CFG
RDR-Password XYZ_RDR
Protect Project Y/N: N

Logfile in $C_CAPE_ALL_ADM_LOG: inspro_XYZ.log

Inserts Project / Press <Accept/Save> if ready
Count: *0 <OSC><DBG> <Replace>

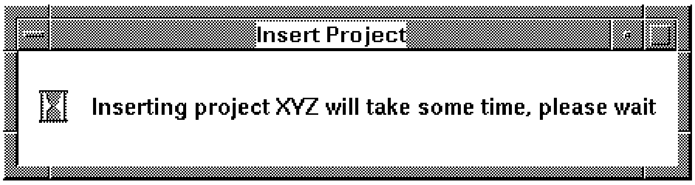
Figure 3-45. Insert Project Dialog Box

CAUTION

DO NOT use the Close button at the top left side of the Administration windows to close the window. This may abort the currently running program and therefore leave the database in an inconsistent state!

If you want to iconize the window, use the Motif *Minimize* function.

After you enter the required data, press **Accept/Save** in the dialog box. The dialog box remains visible and a message box is displayed to indicate the selected function is beginning.
An example message for menu item **Projects Edit Insert** is shown in Figure 3-46.

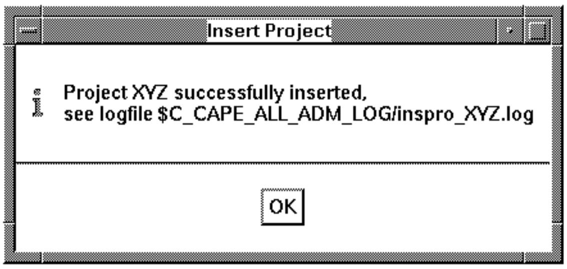


Inserting project XYZ will take some time, please wait

Figure 3-46. Insert Project Message Box

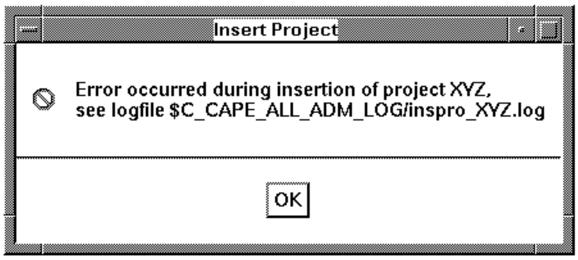
When program execution is finished, that message box is removed and another message box is displayed to indicate the success or failure of the program. Two examples are shown in Figure 3-47 (successful function) and Figure 3-48 (error message). These message boxes have to be acknowledged by clicking **OK**. If you do not acknowledge the message, the logfile might not be complete and it will not be possible to move the cursor in the Administration dialog box.

If program execution is successful, the Administration dialog box is removed and either the Projects or the Users window is re-opened. If program execution is unsuccessful, the respective dialog box remains.



Project XYZ successfully inserted,
see logfile $C_CAPE_ALL_ADM_LOG/inspro_XYZ.log

Figure 3-47. Administration Information Box, Successful Program Execution



Error occurred during insertion of project XYZ,
see logfile $C_CAPE_ALL_ADM_LOG/inspro_XYZ.log

Figure 3-48. Administration Information Box, Unsuccessful Program Execution

Administration Logfiles

All administration functions produce one or more logfiles of the type *<function_label>_<project_or_user>.log.* It is strongly recommended that you review the logfiles after you finish an administration function, especially if an error occurred during program execution.

All logfiles reside in the same directory pointed to by the environment variable $*C_CAPE_ALL_ADM_LOG*. The logfile names are shown at the bottom of the respective Administration dialog boxes.

Table 3-3 shows the logfile names for each Administration menu item.

Table 3-3. Logfile names for Administration Menu Items

Menu Item	Function	Logfile Name	Additional Logfiles
Projects File Print	print project details	print_projects.log	none
Projects File Save	save project release	savpro_<project>_<release>.log	check_dir_<project>.log cop_subtree_<project>_<release>.log
Projects File Restore	restore project release	rstpro_<project>_<release>.log	rst_subtree_<project>_<release>.log
Projects Edit Insert	create new project	inspro_<project>.log	cop_files_<project>.log credir_<project>.log
Projects Edit Delete	delete project	delpro_<project>.log	deldir_<project>.log
Projects Edit Size	enlarge project	enlsiz_<project>.log	check_dir_<project>.log
Projects Edit Modify	modify project	modpro_<project>.log	none
Users File Print	print user details	print_users.log	none
Users Edit Insert	add new user	insusr_<user>.log	credir_<user>.log
Users Edit Delete	delete user	delusr_<user>.log	deldir_<project>.log
Users Edit Password	modify user password	modusr_<user>.log	none

For more complicated administration functions like inserting a new project, the logfile lists all the steps taken by the respective program and concludes with a message such as:

'*<project_function> successfully terminated*' or

'*<project_function> unsuccessfully terminated*'.

If the function fails, the error is specified in more detail, preceded by

'*%ERROR*:' or '*%ERR:*',

Searching the logfile for the percentage sign '%' will help you locate the point of error faster.

In addition to *<function_label>_<project_or_user>**.log***, most Administration tasks also produce a logfile name *<function_label>_<project_or_user>**.txt*** which is for internal use only in case of program failure.

NOTE

As logfiles are not removed automatically, it is advisable to remove old logfiles in $*C_CAPE_ALL_ADM_LOG* from time to time. This can be done by the HP-UX administrator user (default *cape*) only.

NOTE

In cases of failure during execution of Administration programs please read the logfile carefully. If you have to contact customer support, please send in the file *<function_label>_<project_or_user>**.txt***.

Concurrence Of Conflicting Administration Functions

It is necessary to prevent certain administration functions that contradict each other from executing simultaneously. For example, it does not make sense to save a project that another user is currently working on. Also you should not be allowed to delete a project which is being copied to create a new project.

To prevent conflicting functions from executing at the same time, each project has a status showing if an administration function is being performed on this project. All Administration programs first check the status of the project they are supposed to be working on and abort if the project status conflicts with the operation to be performed.

Project status is indicated in the Status column of the Projects window. The possible values are listed in Table 3-4.

Table 3-4. Project Status

Value	Description
G	project is being **G**enerated via menu item **Projects Edit Insert**
C	project is being **C**opied from as a result of a new project being generated as copy of this project via menu item **Projects Edit Insert**
D	project is being **D**eleted via **Projects Edit Delete**
S	project is being **S**aved via **Projects File Save**
R	project is being **R**estored via **Projects File Restore**
E	project is being **E**nlarged via **Projects Edit Size**
M	project is being **M**odified via **Projects Edit Modify**
A	project is **A**ccessible, no manipulations via Administration

When you choose Administration menu items for projects, the respective program will:

- check if any users are working on the project in question (in the following table this is referred to as project status 'work' and it is assumed that the project under work is being modified.
- check if any table locks exist if a project is to be deleted (this is referred to as status 'locked' and is checked explicitly for the action 'Delete Project' as during project deletion all project tables are to be dropped first, which will not succeed if locks exist on the table to be dropped).
- check the project status (see above) and act accordingly. The action depends on the selected function and status of the project as shown in Table 3-5.

Table 3-5. Administration Program Actions

Menu Item	Project Status	Action
Projects **File Save**	work	abort program (project is being modified)
	G	abort program (project not yet complete)
	D	abort program (project is being deleted)
	R	abort program (project in inconsistent state)
	C, S, E, M, A	continue program to save project release
Projects **File Restore**	work	abort program (project is being modified)
	G	abort program (project not yet complete)
	C	abort program (project needed as is for copy)
	D	abort program (don't restore project being deleted)
	S	abort program (project needed as is for save)
	R	abort program (project already being restored)
	E, A	continue program to restore project release
Projects **Edit Insert** (status = status of source project)	work	abort program (source project is being modified)
	G	abort program (source project not yet generated)
	D	abort program (source project is being deleted)
	R	abort program (source project is being restored)
	C, S, E, M, A	continue program to insert new project
Projects **Edit Delete**	work	abort program (project is being modified)
	G,D,R,C,S,E	abort program
	A	continue program to delete project

In addition, two projects of the same name or two users with the same name cannot be created at the same time. Project and user names have to be unique.

This is implemented by using a so-called process file of the type *oma_crepro_<project_name>* and *oma_insusr_<user_name>* in the directory the environment variable *C_CAPE_ALL_ADM_LOG* points to. The file exists as long as the respective project or user insertion program is executed and is removed when the program is terminated.

NOTE

Never remove the files *oma_crepro_<project_name>* and *oma_insusr_<user_name>* in directory $*C_CAPE_ALL_ADM_LOG*.

3.4.1.2 How to Start the Administration Function

To start the Administration function, log in to HP-UX with the administration-level HP-UX user name and password as described in Section 3.1, Application Start-up. When the project log in dialog box is displayed, enter the Administrator username and password, and then click on **Login**. This opens the Administration Window, Figure 3-49. The **Administer** menu in the Administration window has two menu items. The **Projects** menu item displays the Projects window. This window provides the means to manage existing projects, and to create new projects. The **Users** menu item displays the Users window. This window provides the means to manage existing users and assign new users to a specific project.

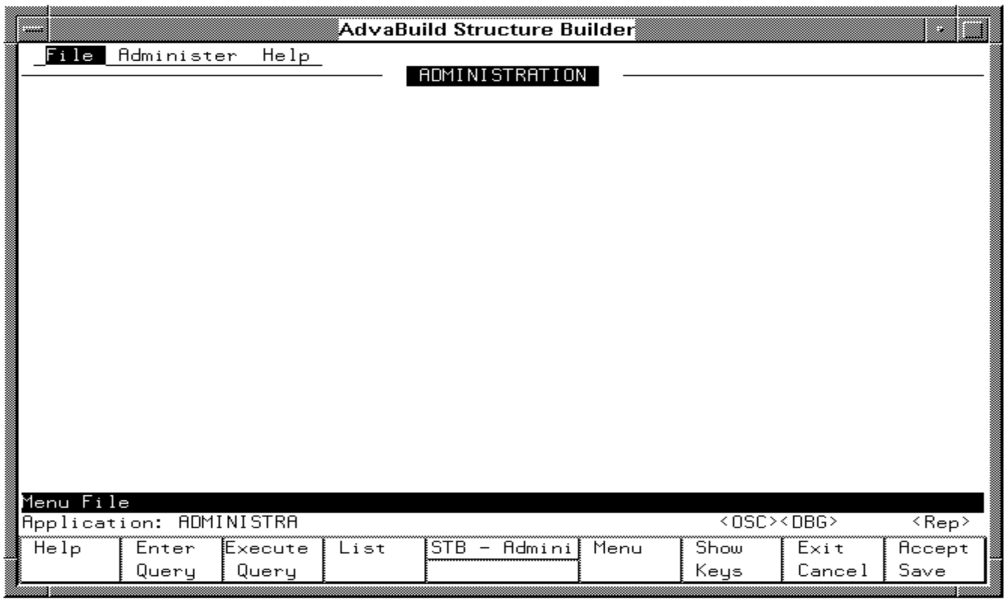


Figure 3-49. Administration Window

NOTE

Only the HP-UX user who owns the Oracle executables (default *cape*) can start Administration.This user must log in to the Structure Builder as user *admin*.

3.4.1.3 How to Create a New Project

The ability to create new projects and assign project users is supported by the AdvaBuild Project Administration function. This tool is intended for use by the database or project manager (administrator) who should understand the Advant Station file and operating system.

AdvaBuild is shipped with a standard project installed. The standard project is the source used as the basis for new projects. It provides all basic data in the database that is needed for a project. If a change is made to the standard project, the change is copied to every new project.

CAUTION

Do not modify the standard project.

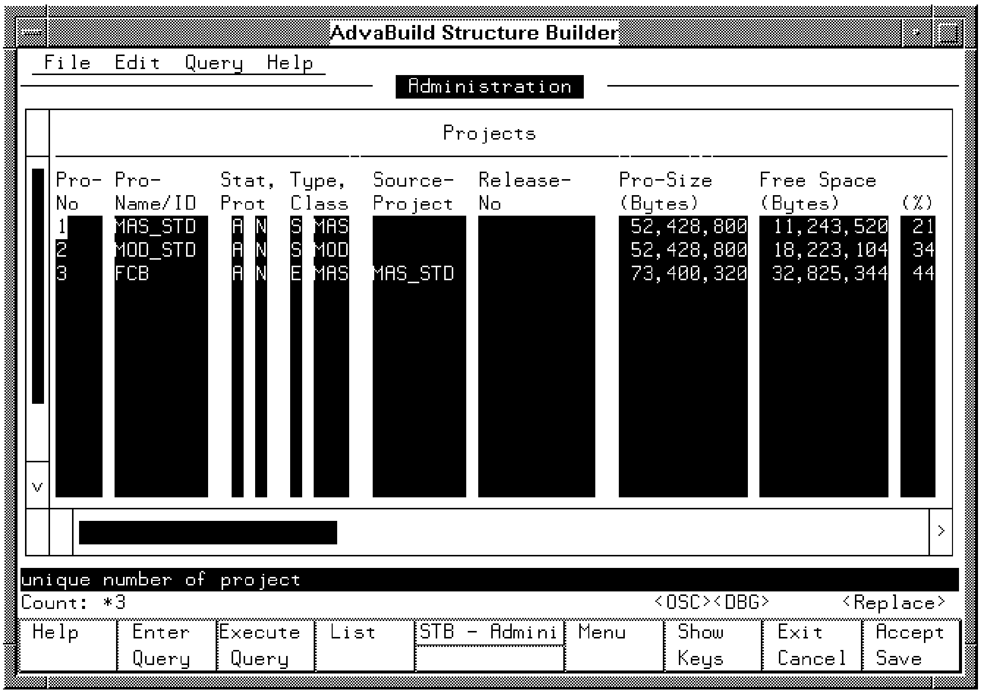
Other projects that you create as copies of the standard project can also be used as the basis for new projects. The following steps are involved in inserting a project:

- A directory subtree for the project is established.
- Space is reserved for the project in the database.
- Two users are inserted for the new project.
- All database and file objects of the source project are copied.

To create a project:

1. Log in to the Administration application as user *admin/admin.*
2. When the Administration window is opened, select the **Menu** function (or press <**f5**>) to access the menu bar.
3. Choose **Projects** from the **Administer** menu.

   This opens the Projects window, Figure 3-50.



Pro- No	Pro- Name/ID	Stat, Prot		Type, Class		Source- Project	Release- No	Pro-Size (Bytes)	Free Space (Bytes)	(%)
1	MAS_STD	A	N	S	MAS			52,428,800	11,243,520	21
2	MOD_STD	A	N	S	MOD			52,428,800	18,223,104	34
3	FCB	A	N	E	MAS	MAS_STD		73,400,320	32,825,344	44

```
unique number of project
Count: *3                                  <OSC><DBG>       <Replace>
```

Figure 3-50. Projects Window

4. Select the **Menu** function to access the Projects window menu bar.
5. Choose **Insert** from the **Edit** menu.

   This displays the Insert Project dialog box with the cursor located in the Project Name field, Figure 3-51.

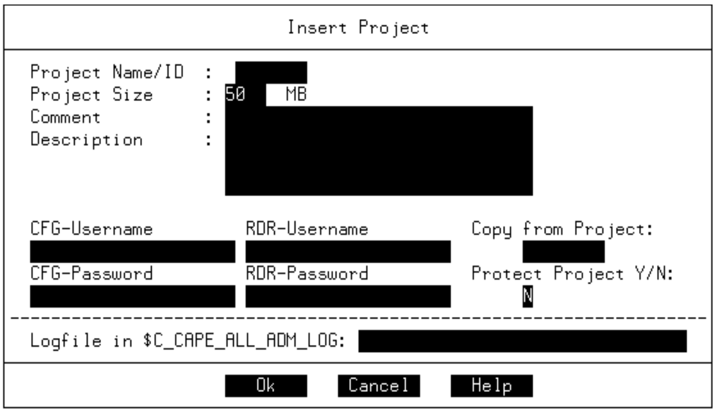


Figure 3-51. Insert Project Dialog Box

6. Enter a unique Project Name (project identification, ID).
7. Press <**Return**> or <**Tab**> to move the cursor to the Project Size field.

   The CFG- and RDR-Username, CFG- and RDR-Password, and Logfile fields are filled in automatically, using the project name.
8. Enter the project size and then press <**Return**>.

   The default size is 50 M. To use the default, simply press <**Return**> or <**Tab**> to move the cursor to the Comment field. If the size you enter is smaller than the source project, the size is adapted automatically after the name of the source project is entered.
9. Enter a project comment if desired.

   The comment is displayed in the topmost object in the Structure Builder tree presentation. This field is optional, if you do not want a comment, simply press <**Return**> or <**Tab**> to move to the next field.
10. Enter a project description (up to three lines) if desired.

    The project description is optional. The project description (if used) is displayed in the footers of document packages generated via the Plant Documentation Builder.

    You may have to press <**Return**> or <**Tab**> three times (once for each line in the Project Description field). This moves the cursor to the CFG-Password field.
11. Define passwords for the project's two default users.

    For each new project, two default users are created: a user <project>_CFG and a user <project>_RDR. The CFG and RDR usernames cannot be changed. Only the passwords can be modified.

    The name and default password for the CFG user is <project_name>_CFG. This user (group 2 = CFG) owns all project tables and can configure the Plant Documentation Builder and ReportWriter. The name and default password for RDR user is <project_name>_RDR. This user has read-only access to the project.

    a. Enter a new CFG password or accept the default by pressing <**Return**>.

       This moves the cursor to the RDR-Password field.

    b. Enter a new RDR password or accept the default by pressing <**Return**>.

       This moves the cursor to the Copy from Project: field.

12. Enter the name of the source project that you want to use as the basis for the new project, and then press <**Return**>.

    You can display a list of valid project names to be used as source project by pressing **List**.

    When the source project has been entered, the size is checked. If the size entered is too small for the source project, the size is adapted to be at least as big.

13. Press **<Return>**, to move the cursor to the Protect Project Y/N field.

14. Protect the project against read-access from other projects if necessary.

    The default entry for the Protect Project field is N = No. This means the project is not protected against read access from other projects.

    If you want to prevent users of other projects from copying parts of your project to their own, choose Y = Yes. You can toggle the protection mode via menu item **Edit Modify** (see Chapter 3.4.1.7, .)

15. Press <**Return**> to move the cursor over the **OK** target at the bottom of the dialog box.

16. Select **Accept/Save**.

    This starts the insert project function and displays a message box. The message asks you to wait (minimum 15 minutes depending on the amount of data to be copied and the load and performance of the machine).

    When the function is finished, this message box is removed and another box is displayed. This message indicates whether or not the function was successful. This message has to be acknowledged by clicking **OK**.

    In either case the logfile provides a detailed description of the function. The logfile *inspro_<project>.log* resides in the directory pointed to by the environment variable *$C_CAPE_ALL_ADM_LOG*.

If the project has been successfully inserted, close the Projects window and insert users as described in Section 3.4.1.4, How to Insert a User.

3.4.1.4 How to Insert a User

When you create a project, two default users are assigned automatically: a configuration user <project>_CFG of user group 2 = CONFIG with special access rights, and a reader user <project>_RDR of user group 4 = READER with read-only access to the project.

The configuration user has unrestricted access to all project data and can configure the Plant Documentation Builder and ReportWriter.

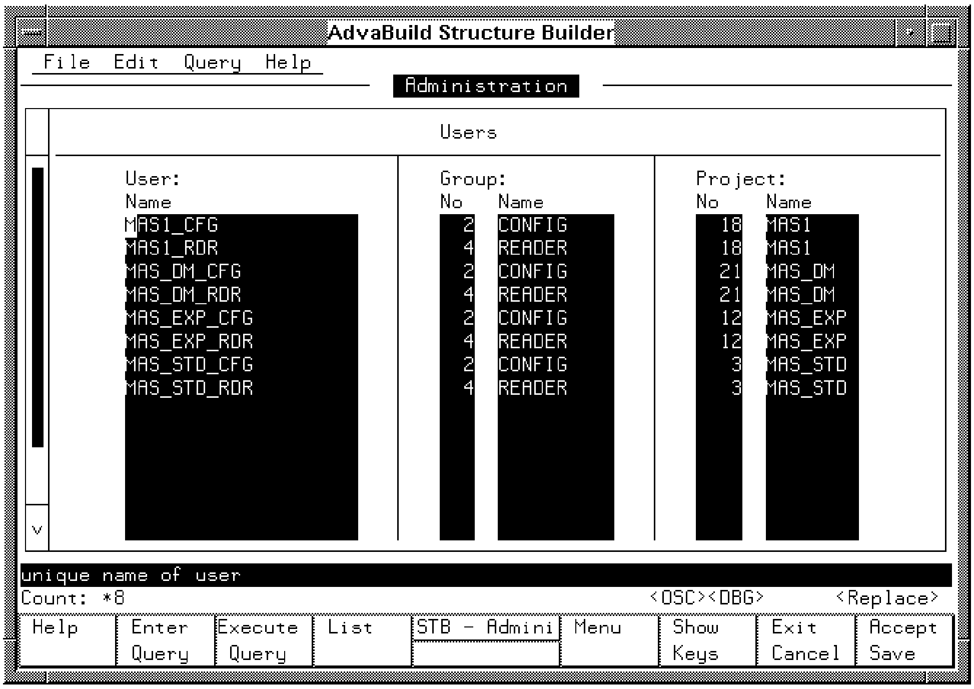
You can assign writer users (user group 3 = WRITER) in addition to the default users to do most of the actual database configuration work.

To insert a user:

1. Log in to the Administration function if you are not already logged in.

2. Choose **User** from the **Administer** menu in the Administration window.

   This opens the Users window, Figure 3-52.



User: Name	Group: No	Name	Project: No	Name
MAS1_CFG	2	CONFIG	18	MAS1
MAS1_RDR	4	READER	18	MAS1
MAS_DM_CFG	2	CONFIG	21	MAS_DM
MAS_DM_RDR	4	READER	21	MAS_DM
MAS_EXP_CFG	2	CONFIG	12	MAS_EXP
MAS_EXP_RDR	4	READER	12	MAS_EXP
MAS_STD_CFG	2	CONFIG	3	MAS_STD
MAS_STD_RDR	4	READER	3	MAS_STD

```
unique name of user
Count: *8                                           <OSC><DBG>      <Replace>
```

Figure 3-52. Users Window

3. Select the **Menu** function to access the Users window menu bar.
4. Choose **Insert** from the **Edit** menu.

   This displays the Insert User dialog box, Figure 3-53.
5. Enter (a unique) user name and press <**Return**>.

   The logfile name is filled in automatically, using the user name.
6. Enter the password and press <**Return**>.

NOTE

User name and password should start with a letter and continue with letters, digits or '_' (underscore). Do not use any special characters like @, # or +.

7. Enter the appropriate group number to establish the access level for the user.

   Usually, group 3 = WRITER is chosen for a normal project user, but you can also select 4 = READER to create a user with read-only access or 2 = CONFIG for a user with special configuration privileges. The corresponding group name is entered automatically.
8. Assign the user to a specific project.
   a. Press <**Return**> to move to the next field.
   b. Enter the project number (as shown in the Pro No column in the Projects window)
   c. Press <**Return**>.

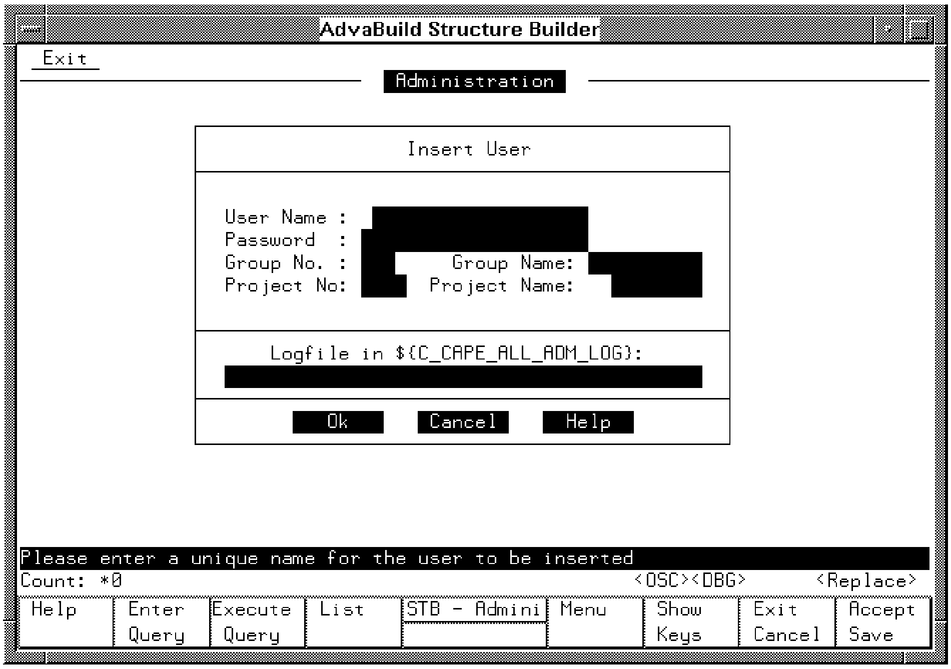
      The corresponding project name is entered automatically, and the cursor is moved to the **OK** target on the bottom of the dialog box.

9. Select **Accept/Save**.

   This starts the insert user function and displays a message box. The message asks you to wait for the function to complete. When the function is finished, this box is removed and another box is displayed. The new box indicates whether or not the function was successful. This message has to be acknowledged by clicking **OK**.

   In either case the log file provides a detailed description of the insertion process. The log file *insusr_<user>.log* resides in the directory pointed to by the environment variable *$C_CAPE_ALL_ADM_LOG*.

10. Repeat this procedure for as many users as required.



```
Please enter a unique name for the user to be inserted
Count: *0                                     <OSC><DBG>      <Replace>
```

Figure 3-53. Insert Users Dialog Box

3.4.1.5 How to Save a Project Release

You can freeze a project release by placing it in a different directory. Project release refers not only to the project data in the AdvaBuild database, but also the operating system objects such as text files, reports and so on. The complete project subtree is copied underneath the user chosen directory. Using this feature has two advantages:

- The amount of data copied is less than the amount copied by the operating system.
- The database does not have to be dismounted for the copy process.

NOTE

Do not copy or restore the database files by operating system commands, especially not before you dismount the database. Simply replacing database files results in an inconsistent database. If you do so before you dismount the database, a database crash occurs.

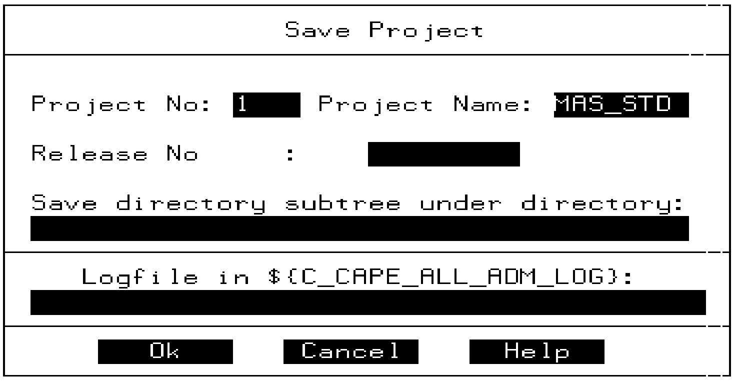
The function for saving a project works as follows:

- The database contents are exported into a single file placed in the project directory *sav* using the Oracle Export utility.
- All project directories except *dat* (which contains the database files) are copied underneath the user chosen directory and the original export file in project directory *sav* is removed.

To save a project:

1. Log in to the Administration application as user *admin*.
2. When the Administration window is opened, select the **Menu** function (or press <**f5**>) to access the menu bar.
3. Choose **Projects** from the **Administer** menu to open the Projects window, Figure 3-50.
4. Select the **Menu** function to access the Projects window menu bar.
5. Choose **Save** from the **File** menu.

   This displays the Save Project dialog box, Figure 3-54.



Save Project

Project No: 1 Project Name: MAS_STD

Release No :

Save directory subtree under directory:

Logfile in ${C_CAPE_ALL_ADM_LOG}:

Figure 3-54. Save Project Dialog Box

6. Enter the number of the project to be saved (the number can be chosen from a list).

   After entering the number, the project name is entered automatically.
7. Press <**Retur**n> or <Tab> to move to field Release No.
8. In the Release No field, enter a unique release ID consisting of characters or numbers.
9. Press <**Retur**n> or **<Tab>** to move to the field containing the name of the directory under which the project directory structure will be saved.
10. Specify the directory.

    To accept the default directory (entered automatically in step 6), go to step 11.

    To choose a directory on another disc, give the complete pathname, for example, */disc2/dir_1/dir_2*.
11. Press <**Return**> to move the cursor over the **OK** target at the bottom of the dialog box.

NOTE

Make sure that write access is granted to the target directory (to the directory path above) in case the target directory has to be created first.

12. Select **Accept/Save**.

    This starts the save project function and displays a message box. The message asks you to wait for the function to complete. When the function is finished, this message box is removed and another box is displayed. This box indicates whether or not the function was successful. This message has to be acknowledged by clicking **OK**.

    In either case the logfile provides a detailed description of the insertion process. The logfile named *savpro_<project>_<release>.log* resides in the directory pointed to by the environment variable *$C_CAPE_ALL_ADM_LOG*.

NOTE

You should check the logfile savpro_<project>_<release>.**txt** as well, especially in case of program error. This logfile provides more detailed information.

You can not restore a project release via the **File Restore** menu item until it has been saved.

3.4.1.6 How to Restore a Project Release

The **File Restore** menu item is used to restore a project release that was saved via **File Save.**

NOTE

A project release can only be restored to the project in the current database it has been saved from. After restoring a project release, all modifications done to the project since the release was drawn are lost. If you want to keep these modifications, save a new release before restoring an old one.

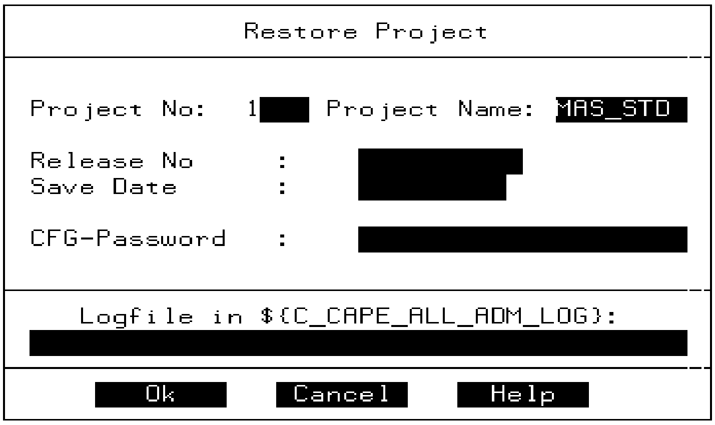
The restore project function works as follows:

- First all project directories except *dat* (containing the database files) and *usr* (containing the user directories) are overwritten by the contents of the saved project directories.
- The contents of the project are overwritten by the contents of the database as it existed during project save. This is done by filling the project tables with the data imported from the export file which was created with the Oracle export utility during **Edit Save**.

To restore a project:

1. Log in to the Administration application as user *admin*.
2. When the Administration window is opened, select the **Menu** function (or press <**f5**>) to access the menu bar.
3. Choose **Projects** from the **Administer** menu to open the Projects window, Figure 3-50.
4. Select the **Menu** function to access the Projects window menu bar.
5. Choose **Restore** from the **File** menu.

This displays the Restore Project dialog box, Figure 3-55.



Restore Project

Project No: 1 Project Name: MAS_STD

Release No :
Save Date :

CFG-Password :

Logfile in ${C_CAPE_ALL_ADM_LOG}:

Figure 3-55. Restore Project Dialog Box

6. Enter the number of the project to be restored (the number can be chosen from a list).

   The project name is entered automatically.

7. Press **<Retur**n> or **<Tab>** to move to field Release No.
8. In Release No field, choose the identifier of the release you want to restore (can be chosen from a list).

   After you enter the release number, the Save Date and Logfile fields are filled in automatically.

9. Press **<Retur**n> or **<Tab>** to move to the field CFG-Password.
10. Enter the configuration user password for the project.

    The configuration user is one of the two users that were created when the project was inserted. The name is <project_name>_CFG. If you have not changed the default password, the password is <project_name>_CFG as well.

11. Press <**Return**> to move the cursor over the **OK** target at the bottom of the dialog box.
12. Select **Accept/Save**.

    This starts the project restore function and displays a message box. The message asks you to wait for the function to complete. When the function is finished, this box is removed and another box is displayed. The new box indicates whether or not the function was successful and has to be acknowledged by clicking **OK**.

    In either case the logfile provides a detailed description of the restoration process. The log file named *rstpro_<project>_<release>.log* resides in the directory pointed to by the environment variable *$C_CAPE_ALL_ADM_LOG*.

NOTES

It is recommended that you check the logfile rstpro_<project>_<release>**.txt** as well, especially in case of program error. This logfile provides more detailed data import information. Search for string '*IMP-*' to find possible import errors.

If an error occurs when you import project data, the program aborts and the database state is inconsistent. In this case start the restoration program again.

3.4.1.7 How to Modify a Project's Comment and Protection

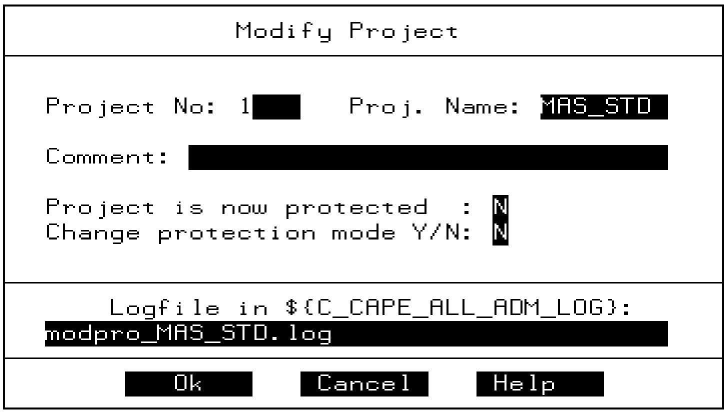
The menu item **File Modify** of the Projects window menu can be used to:

- modify a project's comment and / or
- toggle a project's protection mode.

To modify a project:

1. Log in to the Administration application as user *admin*.
2. When the Administration window is opened, select the **Menu** function (or press <**f5**>) to access the menu bar.
3. Choose **Projects** from the **Administer** menu to open the Projects window, Figure 3-50.
4. Select the **Menu** function to access the Projects window menu bar.
5. Choose **Modify** from the **Edit** menu.

   This displays the Modify Project dialog box, Figure 3-56.



Project No: 1   Proj. Name: MAS_STD

Comment:

Project is now protected : N
Change protection mode Y/N: N

Logfile in ${C_CAPE_ALL_ADM_LOG}:
modpro_MAS_STD.log

Figure 3-56. Modify Project Dialog Box

6. Enter the number of the project to be modified (the number can be chosen from a list).

   After you enter the number, the project name is entered automatically.
7. Press **<Retur**n> or **<Tab>** to move to the Comment field.
8. An entry in field Comment is optional. If the project's comment is not to be modified, simply press **<Return>** or **<Tab>** to move the cursor to the next field.
9. Change the protection status if necessary.

   Y means the project is protected against read access, N means it is not.

   In the Change protection mode Y/N field you can enter Y if this protection mode should be toggled or simply press **<Return>** or **<Tab>** to move the cursor over the **OK** target at the bottom of the dialog box if you want to accept the default value N for No.

10. Select **Accept/Save**.

    This starts the modify project function and displays a message box. The message asks you to wait for the function to complete.When the function is finished, this box is removed and another information box is displayed. The new box indicates whether or not the function was successful and has to be acknowledged by clicking **OK**.

    In either case the log file provides a detailed description of the modification process. The log file named *modpro_<project>.log* resides in the directory pointed to by the environment variable *$C_CAPE_ALL_ADM_LOG*.

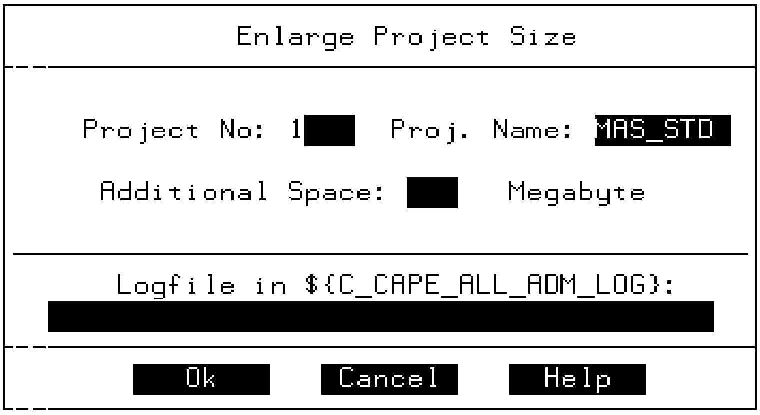
3.4.1.8 How to Enlarge a Project

The menu item **File Size** of the Projects window menu is used to enlarge a project by creating an additional database file. In the Projects window, the total project size in Bytes is shown in column Pro-Size (Bytes), as well as the free space left in Bytes and as percentage of the total project size (columns Free Space (Bytes) and Free Space (%)). It is recommended that you check the free project space from time to time and enlarge a project if there is not much free space left, otherwise an error might occur when trying to insert new objects into the project.

To enlarge a project:

1. Log in to the Administration application as user *admin*.
2. When the Administration window is opened, select the **Menu** function (or press <**f5**>) to access the menu bar.
3. Choose **Projects** from the **Administer** menu to open the Projects window, Figure 3-50.
4. Select the **Menu** function to access the Projects window menu bar.
5. Choose **Size** from the **Edit** menu.

   This displays the Enlarge Project Size dialog box, Figure 3-57.



Enlarge Project Size

Project No: 1 Proj. Name: MAS_STD

Additional Space: Megabyte

Logfile in ${C_CAPE_ALL_ADM_LOG}:

Figure 3-57. Enlarge Project Dialog Box

6. Enter the number of the project to be enlarged (the number can be chosen from a list).

   After entering the number, the project name is entered automatically.

7. Press **<Retur**n> or **<Tab>** to move to field Additional Space.
8. Enter the size in Megabyte by which you want to enlarge the project.
9. Press **<Return>** or **<Tab>** to move the cursor to **OK** and select **Accept/Save**.

   This starts the function to add another database file of the requested size to the project.

   When the program is finished, a message box indicates whether or not the function was successful. This message has to be acknowledged by clicking **OK**.

   The log file named *enlspc_<project>.log* resides in the directory pointed to by the environment variable *$C_CAPE_ALL_ADM_LOG*.

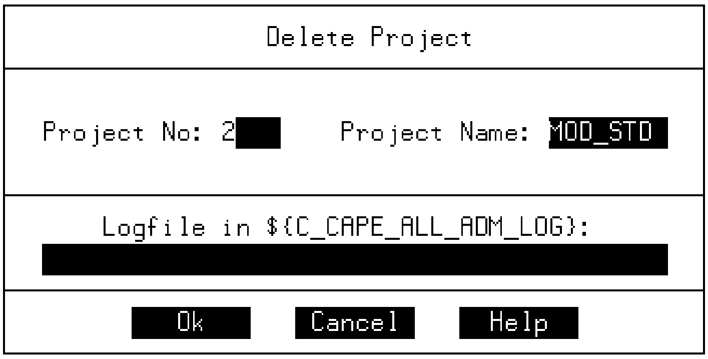
3.4.1.9 How to Delete a Project

The **Edit Delete** menu item of the Projects window is used to delete a project which is not required anymore. The complete project directory subtree including the database files is removed, the project's data is deleted from the database and all users associated with this project are deleted.

To delete a project:

1. Log in to the Administration application as user *admin*.
2. When the Administration window is opened, select the **Menu** function (or press <**f5**>) to access the menu bar.
3. Choose **Projects** from the **Administer** menu to open the Projects window, Figure 3-50.
4. Select the **Menu** function to access the Projects window menu bar.
5. Choose **Delete** from the **Edit** menu.

   This displays the Delete Project dialog box, Figure 3-58.



Delete Project

Project No: 2 Project Name: MOD_STD

Logfile in ${C_CAPE_ALL_ADM_LOG}:

Figure 3-58. Delete Project Dialog Box

6. Enter the number of the project to be deleted (the number can be chosen from a list).

   After you enter the number, the project name is entered automatically.

7. Press **<Retur**n> or **<Tab>** to move to **OK**.

8. Select **Accept/Save**.

   This starts the delete project function and displays a message box. The message asks you to wait for the function to complete. When the function is finished, this box is removed and replaced by another box. The message in this box indicates whether or not the function was successful. Acknowledge by clicking **OK**.

   The logfile named *delpro_<project>.log* resides in the directory pointed to by the environment variable *$C_CAPE_ALL_ADM_LOG*.

NOTE

Standard projects, marked by *S*' in Type column of Projects window, cannot be deleted.

CAUTION

Never delete a project by simply removing the project directory subtree with operating system commands. This will cause the database to crash.

3.4.1.10 How to Handle a Corrupted Project

When you abort the insert project function before it is completed, a corrupted project is created. This can happen when any of the following occur during the insert project function:

- the Insert Project dialog box is closed via the window close button in the top left corner
- the database crashes
- the executing process is killed.

This creates a partial (incomplete) project. As a result, the Projects window still shows the status 'G' for Generated for this project. If you attempt to delete the project, the task fails and an error message is displayed.

%ERROR: project is being generated, aborting!

CAUTION

The project insertion program which usually takes about 15 to 20 minutes might take longer, depending on the workload and the amount of project data to be copied from the source project. Make sure the insert project function for the corresponding project does not execute anymore.

To insert the project again correctly, follow these steps:

First make sure that your project really is corrupted by verifying the insert project function is not executing. Log in as the administration-level HP-UX user (usually *cape*) and type

```
ps -ef | grep oma_crepro
```

The result shows the `grep oma_crepro` command as an executing process. If another process of the type

*cape <pid> <pid> ..... /<pathname>/***oma_crepro <project_name>** *....*

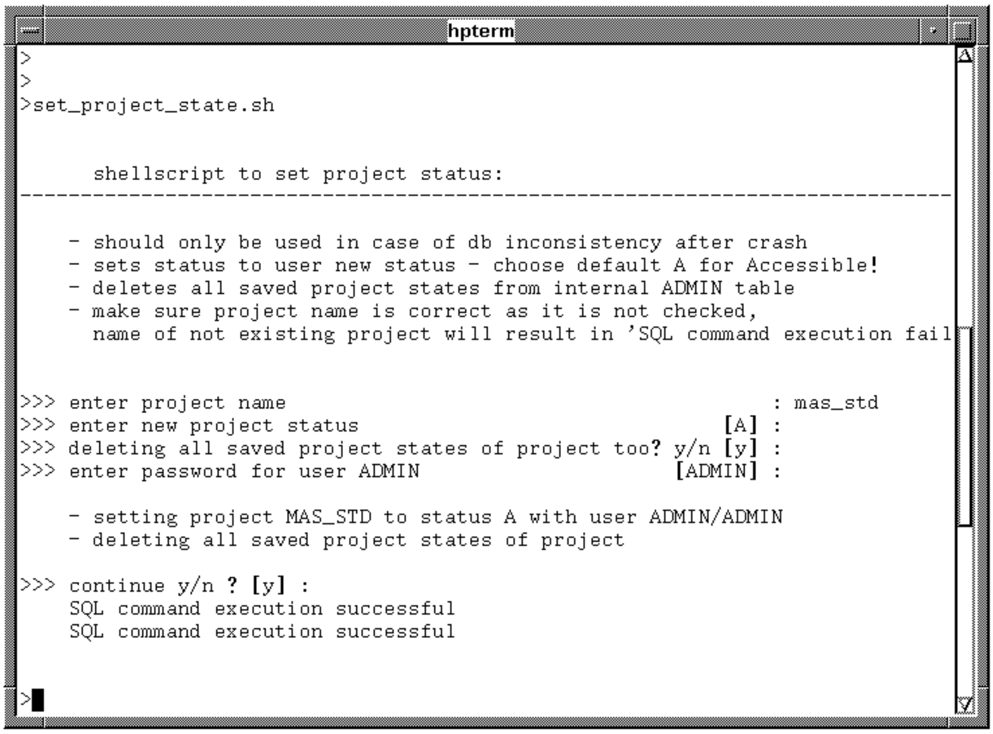
is returned as result where <project_name> is the name of the project in question, the insertion program is still executing and your project is not corrupt. Wait for the program to finish and check the logfile.

Otherwise, enter

```
set_project_state.sh
```

on operating system level. This executes a shellscript to modify a project's status. Enter the corrupted project's name, accept the default project status 'A' by pressing **<Return>** and either accept the default ADMIN password or enter it. Answer to 'continue y/n?' by pressing **<Return>**.

Figure 3-59 shows an example dialog for setting the status of project MAS_STD to 'A'.



```
>
>
>set_project_state.sh


      shellscript to set project status:
------------------------------------------------------------------------------


    - should only be used in case of db inconsistency after crash
    - sets status to user new status - choose default A for Accessible!
    - deletes all saved project states from internal ADMIN table
    - make sure project name is correct as it is not checked,
      name of not existing project will result in 'SQL command execution fail


>>> enter project name                                        : mas_std
>>> enter new project status                              [A] :
>>> deleting all saved project states of project too? y/n [y] :
>>> enter password for user ADMIN                     [ADMIN] :

    - setting project MAS_STD to status A with user ADMIN/ADMIN
    - deleting all saved project states of project

>>> continue y/n ? [y] :
    SQL command execution successful
    SQL command execution successful


>
```

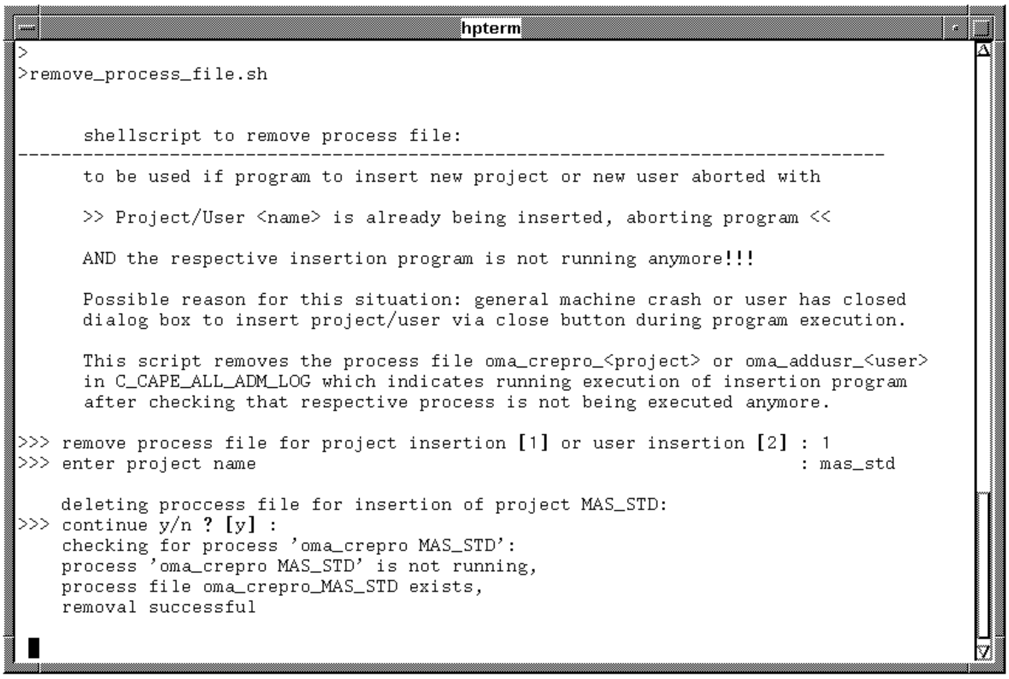
Figure 3-59. Example Dialog for Setting Projects Status to 'A'

Also, the process file oma_crepro_<project_name> in directory $C_CAPE_ALL_ADM_LOG, which indicates execution of the project insertion program for project <project_name>, still exists. To remove it, type

```
remove_process_file.sh.
```

Choose 1 to remove a process file for project creation, enter the corrupted project's name and press **<Return>** to continue.

Figure 3-60 shows an example for project MAS_STD.



```
>
>remove_process_file.sh


      shellscript to remove process file:
--------------------------------------------------------------------------------
      to be used if program to insert new project or new user aborted with

      >> Project/User <name> is already being inserted, aborting program <<

      AND the respective insertion program is not running anymore!!!

      Possible reason for this situation: general machine crash or user has closed
      dialog box to insert project/user via close button during program execution.

      This script removes the process file oma_crepro_<project> or oma_addusr_<user>
      in C_CAPE_ALL_ADM_LOG which indicates running execution of insertion program
      after checking that respective process is not being executed anymore.

>>> remove process file for project insertion [1] or user insertion [2] : 1
>>> enter project name                                                  : mas_std

    deleting proccess file for insertion of project MAS_STD:
>>> continue y/n ? [y] :
    checking for process 'oma_crepro MAS_STD':
    process 'oma_crepro MAS_STD' is not running,
    process file oma_crepro_MAS_STD exists,
    removal successful
```

Figure 3-60. Shellscrip remove_process_file.sh, Example

Now the corrupted project can be deleted by choosing menu item **Edit Delete** from the Projects window. To create the project again, select **Edit Insert** from the Projects window.

3.4.1.11 How to Modify a User’s Password

The **Edit Password** menu item is used to modify a user’s password. To modify a password:

1. Log in to the Administration application as user *admin*.
2. When the Administration window is opened, select the **Menu** function (or press <**f5**>) to access the menu bar.
3. Choose **Users** from the **Administer** menu to open the Users window, Figure 3-52.
4. Select the **Menu** function to access the Users window menu bar.
5. Choose **Password** from the **Edit** menu.

This displays the Modify User dialog box, Figure 3-61.

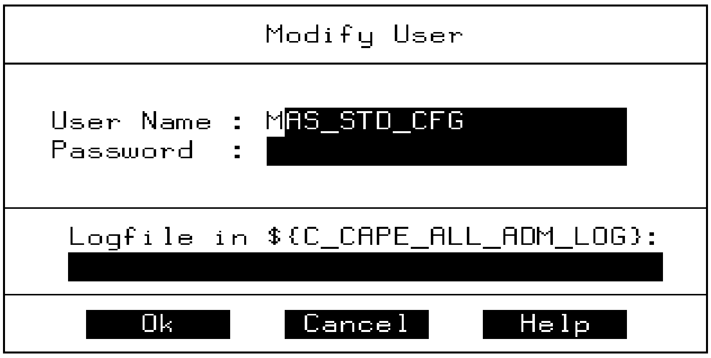


Figure 3-61. Modify User Dialog Box

6. Enter the name of the user whose password is to be modified (can be chosen from a list).
7. Press **<Return**> or **<Tab>** to move to the next field, and enter the new password.
8. Press **<Retur**n> or **<Tab>** to move to **OK**.
9. Select **Accept/Save**.

   This starts the modify password function and displays a message box. The message asks you to wait for the function to complete. When the function is finished, this box is removed and replaced by another box that indicates whether or not the function was successful. Acknowledge by clicking **OK**.

   The logfile named *modusr_<user>.log* resides in the directory pointed to by the environment variable *$C_CAPE_ALL_ADM_LOG*.

3.4.1.12 How to Delete a User

The menu item **Edit Delete** of the Users window menu is used to delete a user which is not required anymore.

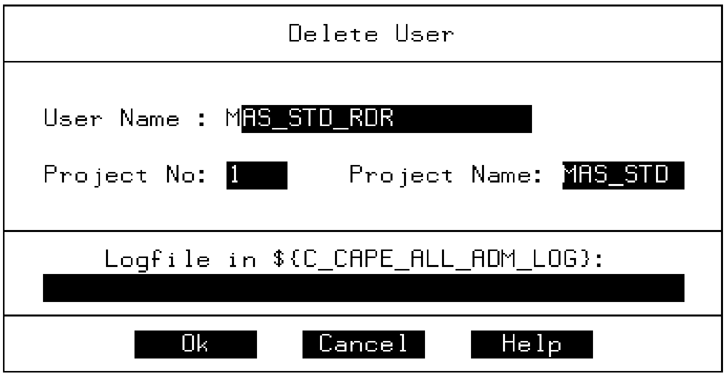
NOTE

The users <project>_CFG cannot be deleted.

To delete a user:

1. Log in to the Administration application as user *admin*.
2. When the Administration window is opened, select the **Menu** function (or press <**f5**>) to access the menu bar.
3. Choose **Users** from the **Administer** menu to open the Users window, Figure 3-52.
4. Select the **Menu** function to access the Users window menu bar.
5. Choose **Delete** from the **Edit** menu.

This displays the Delete User dialog box, Figure 3-62.



User Name : MAS_STD_RDR

Project No: 1 Project Name: MAS_STD

Logfile in ${C_CAPE_ALL_ADM_LOG}:

Figure 3-62. Delete User Dialog Box

6. Enter the name of the user who is to be deleted (can be chosen from a list).

   The project number and name for the user, and the logfile name are filled in automatically.

7. Press **<Return>** or **<Tab>** to move to **OK**, and then select **Accept/Save.**

   This starts the delete project function and displays a message box. The message asks you to wait for the function to complete. When the function is finished, this box is removed and replaced by another box that indicates whether or not the function was successful. Acknowledge by clicking **OK**. The logfile named *delusr_<user>.log* resides in the directory pointed to by the environment variable *$C_CAPE_ALL_ADM_LOG*.

3.4.2 How to Build the Project Structure via the Structure Builder

A project is treated as a hierarchy of objects. To build the project structure, select database objects from a predefined library, and insert, copy, move, and delete them in a hierarchical project structure to meet the functional and documentation requirements for the project.

These functions are performed primarily in the Structure Builder. The Template Builder can also be used to create additional instances of an object type that has already been inserted via the Structure Builder. This is described in Section 3.4.3, How to use the Template Builder.

The application procedures covered in this section include:

- How to open a project in the Structure Builder
- How to navigate in the tree and list presentations
- How to insert and manipulate objects
- How to edit object attributes via the Multiple Template Editor in the Structure Builder
- Miscellaneous message/information functions
  - Error messages
  - Help messages
  - Object Information window.

3.4.2.1 How to Open a Project in the Structure Builder

To open a project in the Structure Builder, log in to HP-UX as described in Section 3.1, Application Start-up. When the Project Login dialog box is displayed, enter the project username (WRITER level or higher) and password defined via the Administration function. This opens the tree presentation of the specified project. When the project is new, the database consists of just the default Project object.

You can also open a project via command line options. For Example:

```
$ stb -un username -pw password -oid objectid &
```

Enter **`stb -help`** for all existing command line options as shown below.

```
 usage: stb [-options ...] where options include:
  help                   print out this message
-debug                   print out debug info
-un username             username login data base
-pw password             password login data base
-oid objectId            object id for setting temp. root
-background color        background color
-bg color                background color
-bordercolor color order color
-bd color                border color
-borderwidth number      border width in pixels
-bw number               border width in pixels
-display displayname     X server to contact
-font fontname           normal text font
-fn fontname             normal text font
-foreground color        foreground color
-fg color                foreground color
-geometry geom           size (in characters) and position
-iconic                  start iconic
-name string             client instance, icon, and title strings
-reverse                 turn on reverse video
-rv                      turn on reverse video
+rv                      turn off reverse video
-selectionTimeout        timeout
-synchronous
-title string            title string
-xnllanguage             language
-xrm resourcestring      additional resource specifications
```

3.4.2.2 How to Navigate in the Tree Presentation

As you insert objects in the project tree structure, the tree may become to large to display in the window in its entirety. Therefore the Structure Builder provides several methods for expanding, compressing, and navigating the tree presentation. One method is via buttons and scroll bars in the tree structure. The other is via the **View** menu (see Section 3.5, Menus).

Subtree Buttons

A subtree button is provided on parent objects that have children objects inserted under them, Figure 3-63. This button provides the means to display or hide the objects under the parent.

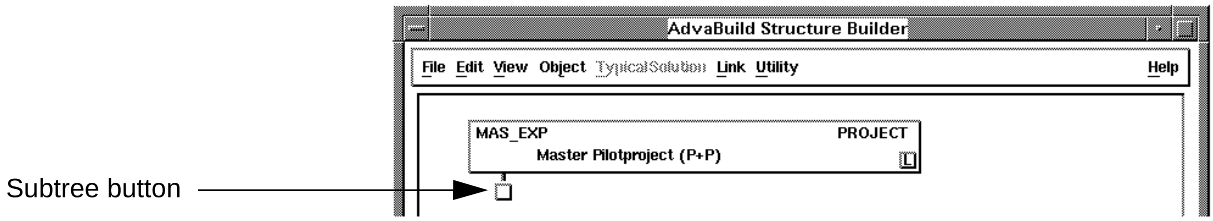

Subtree button
AdvaBuild Structure Builder
File Edit View Object TypicalSolution Link Utility Help
MAS_EXP
PROJECT
Master Pilotproject (P+P)

Figure 3-63. Subtree Button

When the subtree is not displayed below the object, clicking the subtree button displays the next level (direct children of the object). This is equivalent to choosing **Next Level** from the **View** menu. Holding down <**Shift**> and clicking the subtree button displays the whole subtree for the object. This is equivalent to choosing **Expand** from the **View** menu. This is not recommended for very large subtrees because it may be too time-consuming to display the entire subtree. When the subtree is already displayed, clicking the subtree button hides (removes) the subtree. This is equivalent to choosing **Compress** from the **View** menu.

NOTE

No more than 500 objects can be displayed at a time. When you open a subtree that causes you to exceed the limit, the objects over 500 are not displayed and this message is displayed:

```
STB: 184: Cannot display more than 500 tree objects. Reading
is aborted. Use the temporary root function and try again.
```

Parent Button

The parent button is provided when an object is set as a *temporary root*, Figure 3-64.

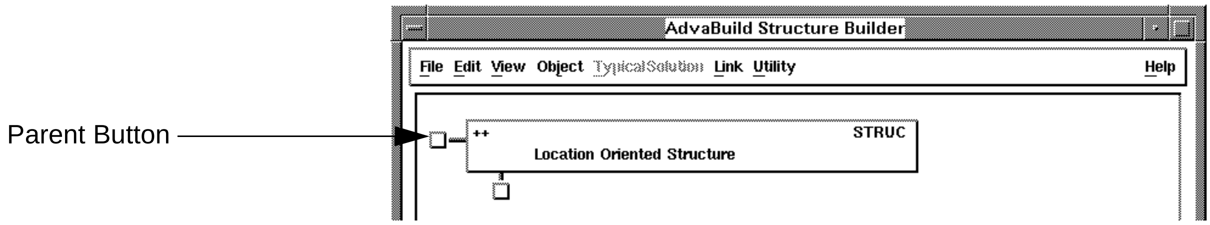

Parent Button
AdvaBuild Structure Builder
File Edit View Object TypicalSolution Link Utility Help
++
STRUC
Location Oriented Structure

Figure 3-64. Parent Button

You can temporarily set any object as the root object. This removes any objects above the selected temporary root, and facilitates navigation within the subtree of the temporary root.

Clicking the parent button once displays the parent object above the temporary root, and its next level (direct children). Thus you can step your way back up to the original root object. Or you can return directly to the original project root by choosing **Project Root** from the **View** menu.

3.4.2.3 How to Access the List Presentation from the Tree Presentation

There are two ways to access the list presentation from the tree presentation. One method is to click on the List button on the applicable object in the tree presentation, Figure 3-65. This button is only on tree objects that have children inserted under them in the list presentation.

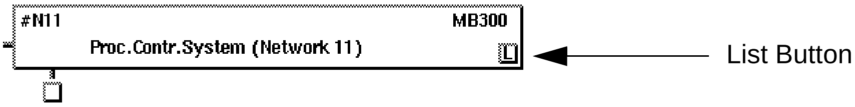


Figure 3-65. List Button

Or you can select the object in the tree presentation whose list objects you want to view, and then choose **List** from the **View** menu. This displays the List of Object dialog box, Figure 3-66.

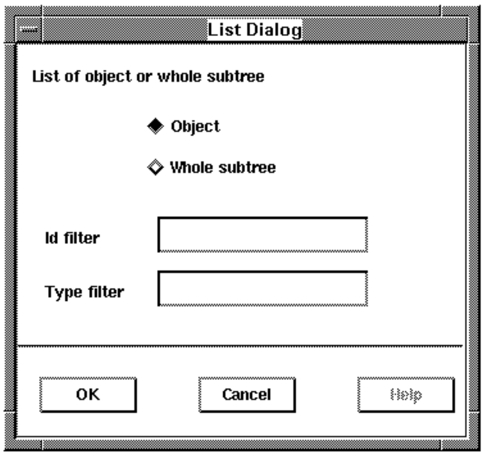


Figure 3-66. List of Object Dialog Box

This dialog box provides greater flexibility for determining specifically which list objects are included in the list presentation. If the Object button is selected (default condition), the list window displays only list objects that are direct children (first level) below the selected tree object. If the Whole subtree button is selected, the list window displays list objects for the whole subtree (all descendents) below the selected tree object.

There are two filters (ID and Type) which can be used to further limit the subset of list objects. Filtering means that only those objects whose ID and/or object type match the specified filters are displayed in the list window. The percent sign "%" and the underscore "_" are wildcard characters. "%" sets the filter to match zero or more characters, and "_" sets the filter to match exactly one character. For example:

- **CS1%** represents CS1 followed by any character string
- **%.PID** represents any character string followed by.PID
- **CS_CM1** represents the string CS_CM1 where the underscore can be any one character.

3.4.2.4 How to Navigate in the List Presentation

The list presentation does not have subtree and parent buttons. Instead you can navigate either by using the scroll bar, or the **Query** menu. To execute a query, choose **Enter** from the **Query** menu. This provides a dialog box where you specify query parameters, Figure 3-67.

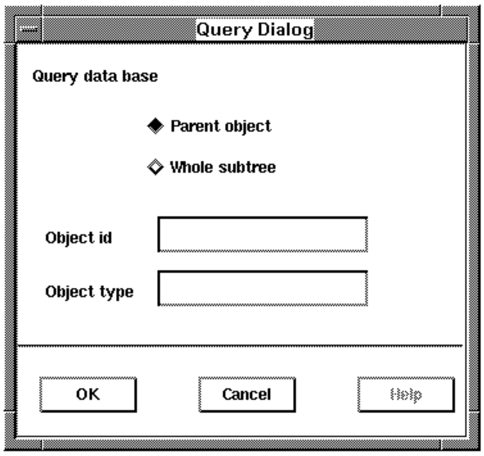


Figure 3-67. Query Dialog Box

Specify whether you want to query children of the specified parent object, or the whole subtree. If the Parent Object button is selected (default), the list window displays list objects that are direct children (first level) of the selected parent object. If the Whole subtree button is selected, the list window displays the whole subtree (all descendents) of the selected parent object.

Object ID and Object Type filters can be used to limit the subset of list objects displayed in the list window. When you use a filter only the objects whose ID and/or object type match the specified filters are displayed in the list window. Percent sign "%" and underscore "_" are wildcard characters. "%" sets the filter to match zero or more characters, and "_" sets the filter to match exactly one character. For example:

- **CS1%** represents CS1 followed by any character string
- **%.PID** represents any character string followed by.PID
- **CS_CM1** represents the character string CS_CM1 where underscore can be any character.

Click **OK** to open a list window with the specified list objects. Queries for a large list of objects may take some time.

3.4.2.5 How to Create New Objects in the Project Structure

There are three methods for creating new objects: insert, copy, and import. These methods are the same for the tree and list presentations. Note that objects inserted in the tree are not displayed in the list, and objects inserted in the list are not displayed in the tree. However, you can remove selected objects from the tree and place them in the list, or remove selected objects from the list and place them in the tree as required. This functionality is provided by the Object Information window and is described in Section 3.4.2.9, Object Information Window.

How to Insert an Object

When you are starting from scratch, or have no suitable objects to copy, you must insert new objects. You can insert objects either via the Structure Builder, or the Template Builder (multiple-object form only). To insert an object via the Structure Builder, click on an existing object where you want to insert a new object in the tree structure, then choose **Insert** from the **Edit** menu. This displays the Insert Object dialog box. This dialog box provides the means to enter a unique object ID, and a comment (optional). In addition this dialog box lets you insert an object either as a child of the selected object, or as a sibling (On the same level). Refer to Section 3.5.5.2, Edit Menu for details.

For information on inserting objects via the Template Builder multiple-object form, refer to Section 3.4.3.4, How to Insert Objects via the Template Builder.

How to Copy Objects

If you have already built a subtree in the project tree structure, and you need to build more subtrees that closely resemble the existing one, you can copy existing objects. When you copy an object, you copy the entire subtree. Copying saves time when you are configuring multiple subtrees that are very similar. Rather than configure each subtree from scratch, you can configure one, then copy it and use it as the basis for other similar subtrees. You can copy objects either via the Structure Builder, or Template Builder (multiple-object form).

To copy an object via the Structure Builder, select the object and then choose **Copy** from the **Edit** menu. This displays the Copy Object dialog box. You can copy either a single object or an object and its entire subtree. The Copy Object dialog box provides the means to apply an Object ID mask to the duplicate object(s) to assure that object IDs for duplicate objects are unique. This dialog box also provides the means to specify a new parent object for the duplicate object(s).

By default, the duplicate object and its subtree are appended to the bottom of the specified parent object's subtree. If execution order is critical, and you need to insert the duplicate object(s) in a specific location of the parent object's subtree, select the On the same level button. This function lets you specify the predecessor object to insert the duplicate object(s) under. When you select On same level, you cannot select a new parent object. The duplicate objects must be inserted in the same subtree as the original object. If execution order is not critical, do not use On same level, especially if you need to copy an object to a different parent.

Refer to Section 3.5.5.2, Edit Menu.

How to Import Objects From Another Project

Import is used to copy an object and its related subtree from another project into the current project. Again, this method can save you time when the new project you are working on is similar to another existing project.To import objects from another project choose **Import** from the **File** menu. This displays the Copy Project dialog box. Refer to Section 3.5.5.1, File Menu for details.

3.4.2.6 How to Move, Delete, and Rename Objects

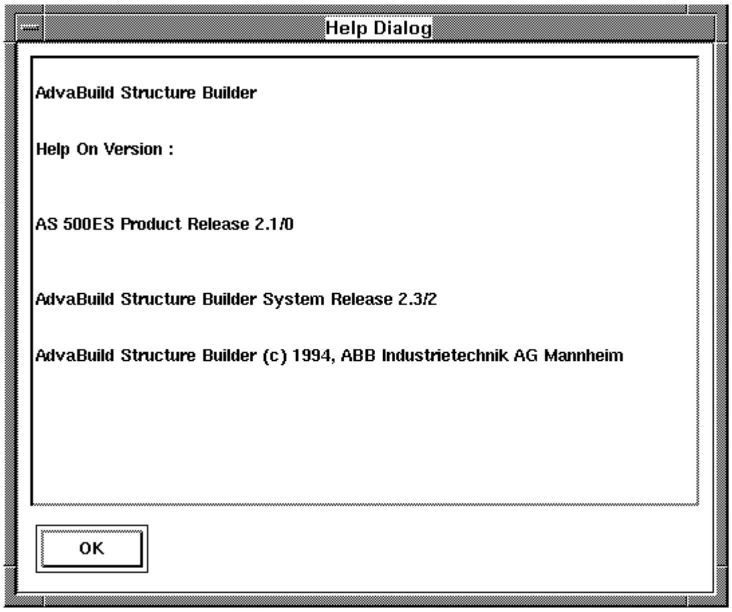
The Structure Builder **Edit** menu provides menu items for moving, deleting, and renaming objects. Refer to Section 3.5.5.2, Edit Menu for details.

In the tree presentation, the object ID and comment can be renamed by clicking in the appropriate area, Figure 3-68. This displays a text field where you can enter the new ID or comment. Save the modification by pressing **<Return>**.

Figure 3-68. Tree Object Renaming

3.4.2.7 Help Window

A typical Help window is shown in Figure 3-69. Scroll bars are provided when help text is longer than can fit in the window. Clicking **OK** removes the Help window.



AdvaBuild Structure Builder

Help On Version :

AS 500ES Product Release 2.1/0

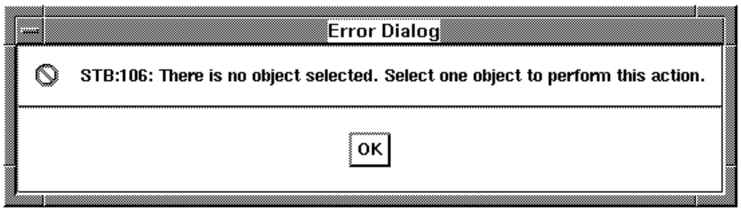
AdvaBuild Structure Builder System Release 2.3/2

AdvaBuild Structure Builder (c) 1994, ABB Industrietechnik AG Mannheim

Figure 3-69. Help Window

3.4.2.8 Error Messages

All messages are displayed as shown in Figure 3-70. Clicking **OK** removes the messages box.

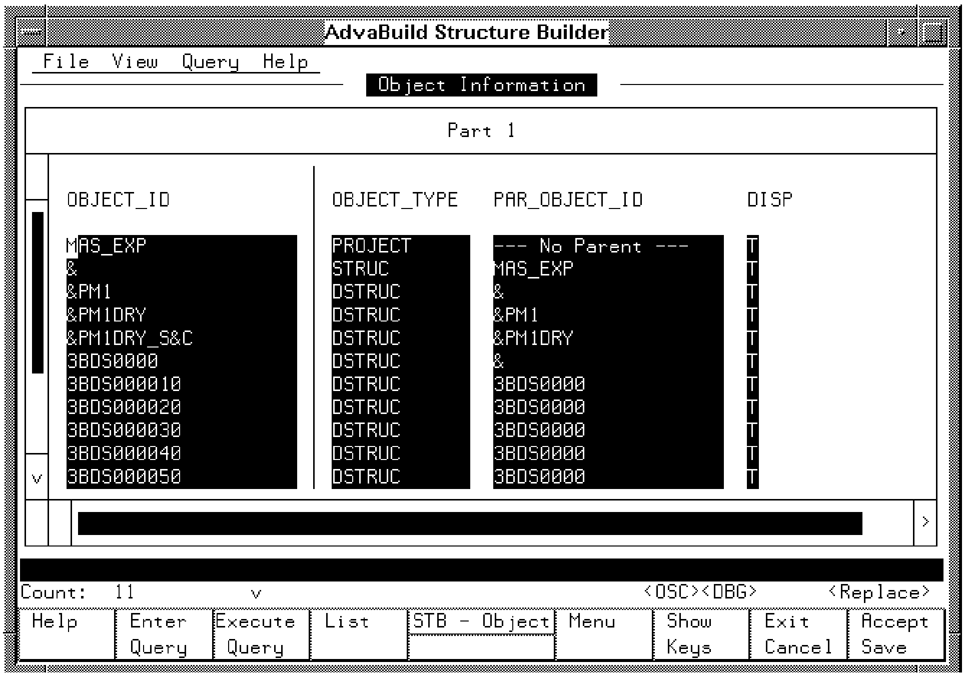


STB:106: There is no object selected. Select one object to perform this action.

Figure 3-70. Error Messages

3.4.2.9 Object Information Window

The Object Information window, Figure 3-71, provides information about objects such as creation date, modification date, loan attributes, and so on. To open this window choose **Information** from the **Object** menu. You can use this window to search for objects in a project or subtree by the **Query** menu, and to change the presentation (tree or list) for objects.
The presentation state is indicated by **T** (tree presentation) or **L** (list presentation).



```
                              Part 1

OBJECT_ID            OBJECT_TYPE   PAR_OBJECT_ID          DISP

MAS_EXP              PROJECT       --- No Parent ---      T
&                    STRUC         MAS_EXP                T
&PM1                 DSTRUC        &                      T
&PM1DRY              DSTRUC        &PM1                   T
&PM1DRY_S&C          DSTRUC        &PM1DRY                T
3BDS0000             DSTRUC        &                      T
3BDS000010           DSTRUC        3BDS0000               T
3BDS000020           DSTRUC        3BDS0000               T
3BDS000030           DSTRUC        3BDS0000               T
3BDS000040           DSTRUC        3BDS0000               T
3BDS000050           DSTRUC        3BDS0000               T

Count:  11          v                          <OSC><DBG>      <Replace>
```

Figure 3-71. Object Information Window

To move an object from one presentation to the other:

1. Select the parent object.
2. Choose **Information** from the **Object** menu.
3. Scroll through the DISP column. If an object is in the tree presentation, and you want to put it in the list presentation enter **L**. If an object is in the list presentation, and you want to put it in the tree presentation enter **T**.
4. Press **Accept/Save** when you are finished.

3.4.3 How to use the Template Builder

The Template Builder provides the means to edit attributes for objects that you insert via the Structure Builder. In addition you can insert objects via the Multiple-object form of the template Builder.

Editing object attributes in the Template Builder involves opening the objects you inserted via the Structure Builder and then editing the object attributes via user-configurable fill-in-the-blank templates. When you work in the Template Builder, you are actually working in a temporary workspace. To record your work permanently in the database you must save it (either via **Save** in the **File** menu, or the **Accept/Save** function key). This arrangement minimizes the likelihood of mistakes. You can make a change in the workspace and then discard it by exiting without saving. If changes are discarded, the database is not affected.

3.4.3.1 How to Start the Template Builder

When you work in the Template Builder, you can work with just a single object at a time, or you can work with multiple objects of a selected type in a given subtree. Note that the menu of the multiple-object form is equal to the menu of the single-object form. The execution of all menu items in both forms results in the same outcome.

Starting the Single Object Form

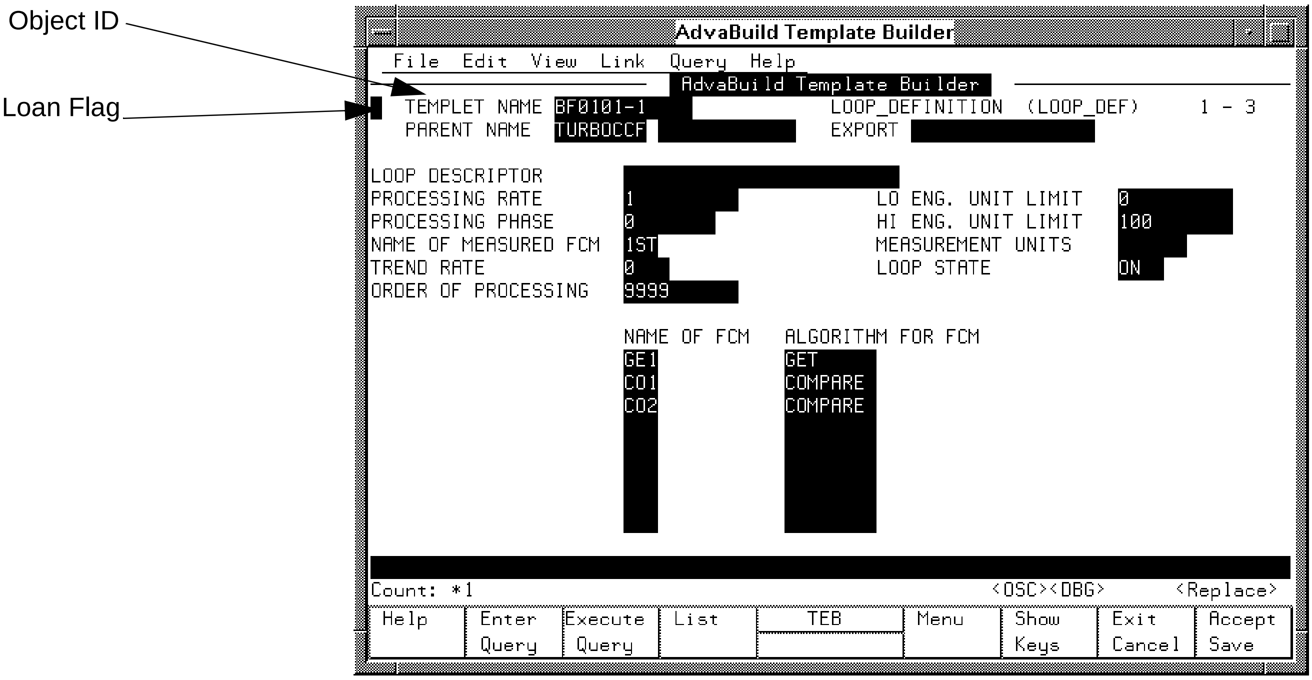
To start the Single-object Form, either double-click on the object that you want to open, or click once and then choose **Open** from the **Object** menu.

This displays a single-object template form, Figure 3-75. This form provides the means to view and modify object attributes. The contents of this display depend on the type of object selected.

NOTE

Some objects have more than one view. In such cases, choosing **Open** displays a dialog box for selecting either the Template Builder view (TEMPLATE) or the alternative view. To open the Template Builder view, select **TEMPLATE**. Selecting another view (for example, **FUNCT_CHART** for the Function Chart Builder view) opens the other view for the selected object.

The Single-object Form displays just one object at a time. If all the attributes cannot be displayed on one page, multiple pages are provided. You can scroll vertically between the fields. Each row contains the name and value of one attribute. On top of the page one row displays the Object-Identification (`OBJ_ID`) and the Loan Flag (`LOAN_FLAG`). When the LOAN column is set to **Y**, the corresponding object is loaned by another tool. If the LOAN column is empty, the object is not loaned by another tool. When an object is loaned by another tool, the object may not be modified with the Template Builder.



```
  TEMPLET NAME BF0101-1                 LOOP_DEFINITION  (LOOP_DEF)      1 - 3
  PARENT NAME  TURBOCCF                 EXPORT

LOOP DESCRIPTOR
PROCESSING RATE       1                     LO ENG. UNIT LIMIT    0
PROCESSING PHASE      0                     HI ENG. UNIT LIMIT    100
NAME OF MEASURED FCM  1ST                   MEASUREMENT UNITS
TREND RATE            0                     LOOP STATE            ON
ORDER OF PROCESSING   9999

                      NAME OF FCM   ALGORITHM FOR FCM
                      GE1           GET
                      CO1           COMPARE
                      CO2           COMPARE

Count: *1                                         <OSC><DBG>       <Replace>
```

Figure 3-75. Template Builder, Single Object Form

Starting the Multiple Object Form

If you already have the single-object form open for the object type, you can change the view to the multiple-object form by choosing **Toggle Form** from the **View** menu on the Template Builder menu bar. Press **Menu** to move the cursor into the menu bar. Then use the right arrow key to select the **View** menu, and press <**Return**> to select **Toggle Form**. You can also use a key function as described in Section 1.9.4.2, Special Key Functions to toggle between the multiple-object form and the single-object form.

To start the Multiple-object form from the Structure Builder, click on the parent object of the subtree where you want to open objects, and then choose **Open Type** from the **Object** menu.

This displays a dialog box for specifying the desired object type, Figure 3-76.

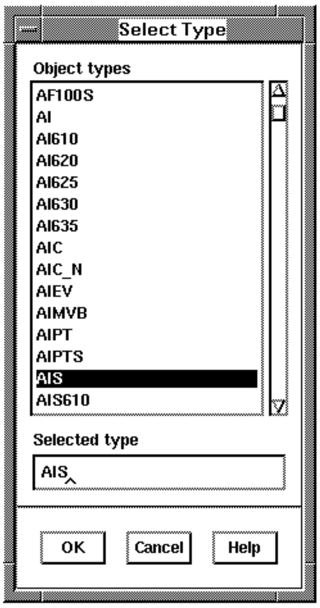


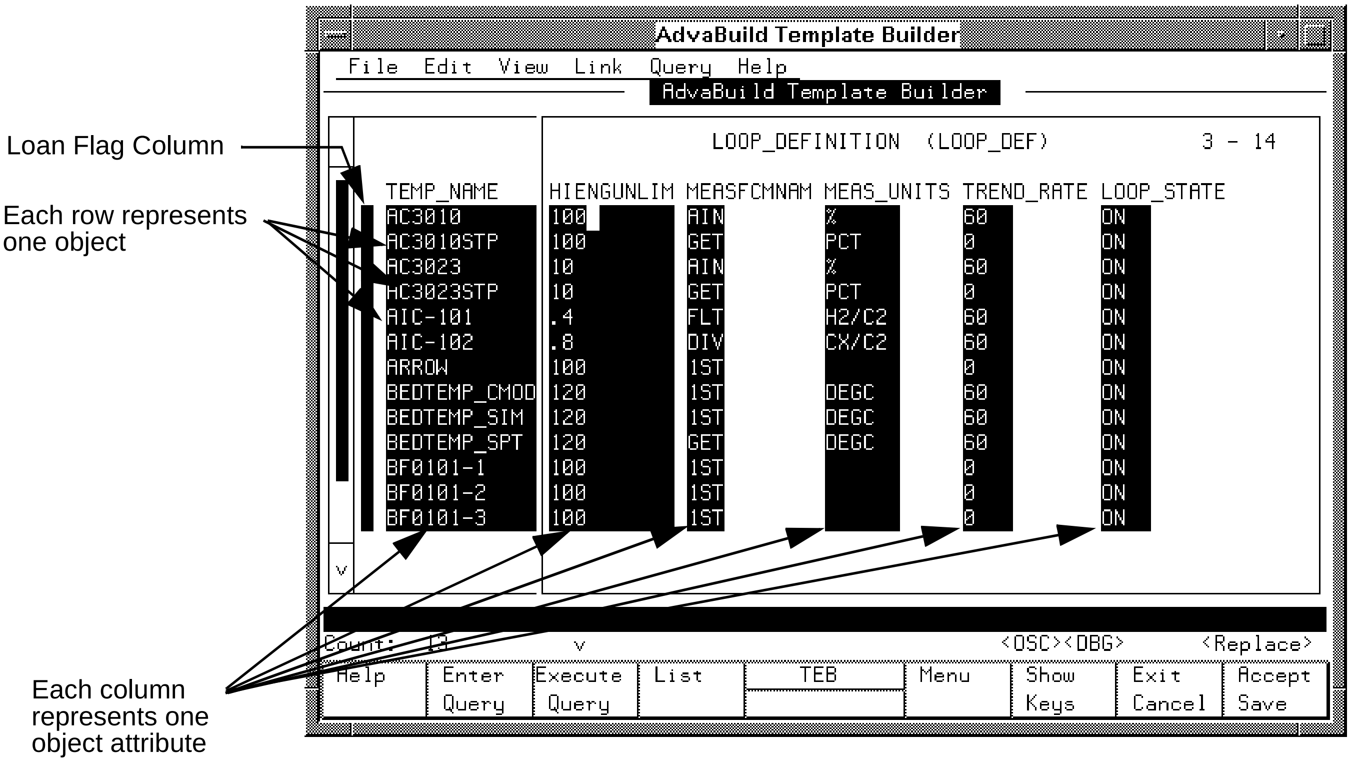
Figure 3-76. Object Types Dialog Box

Select the desired object type and then click on **OK**. This opens the Multiple-object form with a list of all objects of the selected object type that belong to the subtree that you selected. The attributes may be displayed on several pages, depending upon the number of objects and attributes.

On the Multiple-object Form, Figure 3-77, each row represents one object, and each column represents one attribute for all objects included on the form. The attribute name is displayed on top of each column. This name is not necessarily the attribute name in the database.

The columns `OBJ_ID` and `LOAN`, are used in the Oracle database for identification and long-time locking and are displayed on every page.

The `OBJ_ID` must be unique in the system. Therefore, this field is well suited when searching for objects. The OBJ_ID field is not displayed in full length. Only the first twenty characters are displayed on the form. The actual length in the database is 32 characters. You can scroll the OBJ_ID field, or display it fully via the Field Editor **<ESC** >.


Loan Flag Column
Each row represents one object
Each column represents one object attribute
AdvaBuild Template Builder
File
Edit
View
Link
Query
Help
AdvaBuild Template Builder
v
Help
Enter Query
Execute Query
List
TEB
Menu
Show Keys
Exit Cancel
Accept Save

LOOP_DEFINITION (LOOP_DEF) 3 - 14

TEMP_NAME	HIENGUNLIM	MEASFCMNAM	MEAS_UNITS	TREND_RATE	LOOP_STATE
AC3010	100	AIN	%	60	ON
AC3010STP	100	GET	PCT	0	ON
AC3023	10	AIN	%	60	ON
AC3023STP	10	GET	PCT	0	ON
AIC-101	.4	FLT	H2/C2	60	ON
AIC-102	.8	DIV	CX/C2	60	ON
ARROW	100	1ST		0	ON
BEDTEMP_CMOD	120	1ST	DEGC	60	ON
BEDTEMP_SIM	120	1ST	DEGC	60	ON
BEDTEMP_SPT	120	GET	DEGC	60	ON
BF0101-1	100	1ST		0	ON
BF0101-2	100	1ST		0	ON
BF0101-3	100	1ST		0	ON

Count: 13 <OSC><DBG> <Replace>

Figure 3-77. Template View, Multi-object Form

3.4.3.2 How to Save Your Work in the Template Builder

Before you edit objects in the Template Builder, you should know how to save your work. Modifications are not recorded directly in the database; rather, they are held in a temporary workspace. While modifications are held in the workspace, other users cannot see them. Therefore, if one user accesses an object that another user has modified but not saved, the user that is accessing the object sees the unmodified version of the object. To make the modifications permanent in the database, select **Accept/Save** or choose **Save** from the **Edit** menu.

The Template Builder provides automatic locking of objects to prevent two or more users from updating the same object at the same time. When you attempt to update an object, the Template Builder determines whether the object has been updated or deleted by another user since the last time the object was retrieved. If the object has been updated or deleted since the user has retrieved it, you must re-execute the query in order to see and work with the revised version of the object. However, if the object has not been changed by another user, the Template Builder will lock that object so that other users cannot make modifications while one user is updating it.

3.4.3.3 How to Edit Object Attributes via the Template Builder

There are three methods for editing attribute fields on templates: manual typing, List of Values function, and **Copy Fields** from the **Edit** menu.

Manually Typing Entries

This is the basic method for editing attribute fields on templates. When you open an object and display the template form, the cursor is located on the first user-configurable field. The **<Tab>**, **<Return>** and **<Shift-Tab>** keys are used to move the cursor between fields, and the up, down, left, and right arrow keys are used to move the cursor within a field.

The Template Builder provides two modes for typing entries in data fields:

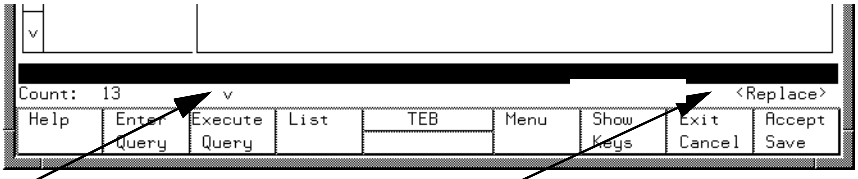
- Insert Mode

  When Insert mode is active, as you enter a character, the character currently at the cursor and all following characters are shifted to the right.

- Replace mode.

  When Replace mode is active, each character entered replaces the character at the cursor. The characters after the cursor do not move.

You can toggle between the Insert and Replace modes by selecting **Insert/Replace** (**<Ctrl-A>**). Once the mode is set, it remains active until changed by pressing the key again. The status line displays the current mode, Figure 3-78.



Status Line

Indicates Replace Mode is active

Figure 3-78. Insert/Replace Indicator

To delete a character, place the cursor on the character and select **Delete Character** (**<Ctrl-D>**). **Delete Character** functions in the same manner whether Insert or Replace mode is active. To delete a character before the cursor, select **Delete Backward** (**<Ctrl-shift-D>**).

List Function

Some attribute fields have a pre-defined list of entries. To display the List of Values window for the current field, Figure 3-79, select **List** (**<f4>**). To select a value in the list, use the arrow keys to highlight the value and select **Accept/Save**. This copies the selected value into the attribute field and closes the List of Values window. To exit the List of Values window without making a selection, select **Exit Cancel.**

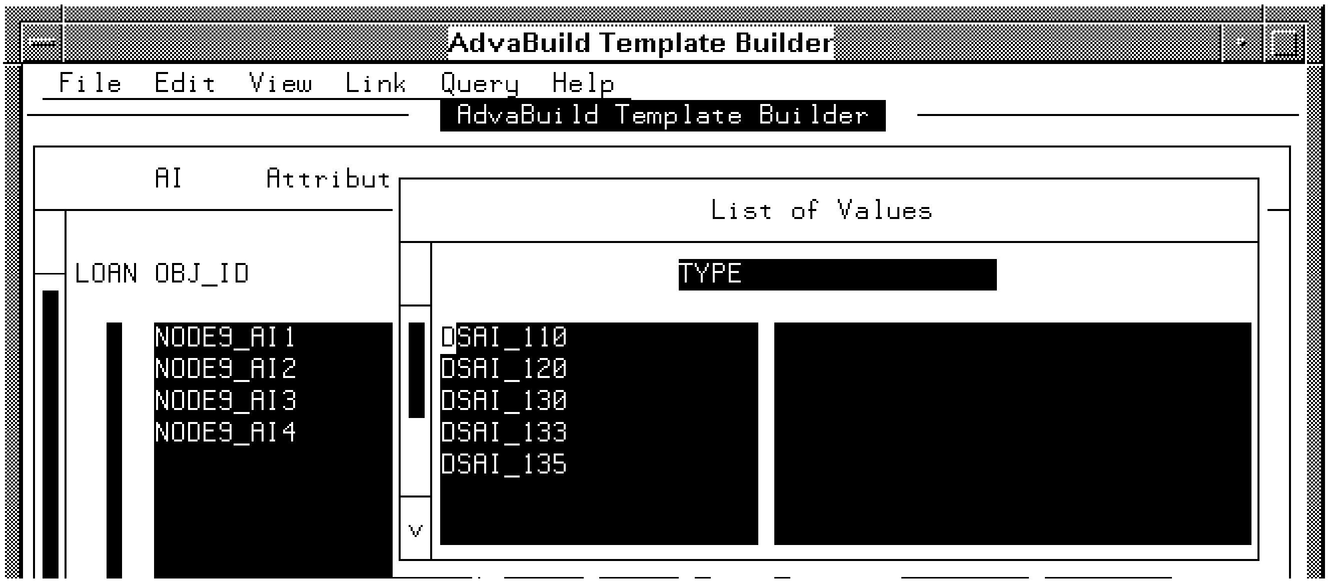


Figure 3-79. List of Values Window

Edit Menu - Copy Field

The **Edit** menu in the Template Builder provides a **Copy Field** item for editing one or more objects at a time. You select the type of attribute to copy by locating the cursor in the attribute field for the object that you want to copy to. The actual value that you are copying is taken from the object above. When you choose **Copy Field**, the following options are provided:

- **Single** - copy value from object above selected object to just a single (selected) object
- **Multiple** - copy value from object above selected object to a specified number of objects, starting at selected object
- **To End** - copy value from object above selected object to an unspecified number of objects starting at selected object and finishing at the end of the form

Refer to Section 3.5.8.2, Edit Menu for details.

3.4.3.4 How to Insert Objects via the Template Builder

You can use either the **Insert** or **Copy** menu items in the **Edit** menu in the Template Builder to insert new objects.

Inserting New Objects in the Multiple Object Form

The **Edit** menu of the Multiple Object form in the Template Builder provides the means to insert objects in the current subtree. To insert an object:

1. Position the cursor on the line for the object that you want to insert the new object under.
2. Click on the **Menu** function.

3. Use the right arrow key to position the cursor on the **Edit** menu and press **<Return>**.
4. Use the down arrow key to position the cursor on **Insert** and press **<Return>** to select.

   This creates a new object below the current cursor position and assigns default values to each attribute. You must enter a unique OBJ_ID for the new object. You can edit the other object attributes as required. After inserting an object, you cannot commit (save) your changes until you have assigned the new object a unique object ID.

You can also use **<Insert line>** to create a new object. Refer to Section 3.5.8.2, Edit Menu.

Copying Objects in the Multiple Object Form

The Edit menu of the Multiple Object form in the Template Builder provides the means to copy objects in the current subtree. To copy an object:

1. Position the cursor on the line for the object you want to insert the new object under.
2. Click on the **Menu** function.
3. Use the right arrow key to position the cursor on the **Edit** menu and press **<Return>**.
4. Use the down arrow key to position the cursor on **Copy** and press **<Return>**.

   This displays the following **Copy** options: **Single**, **Multiple**, and **To End**, Figure 3-80.

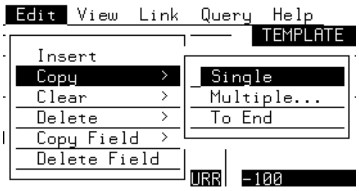


Figure 3-80. Menu item: Edit Copy

5. Select the appropriate option in the **Copy** menu as described below.

   Single creates a new object with the same object ID and attribute values as the current object (including attributes not actually displayed in the form). The new object is assigned to the parent given within dialog box “Set New Parent Object” (refer to menu “View - Set Parent” for details). After copying an object, you cannot commit (save) your changes until you have assigned the new object a unique object ID.

   Multiple creates *n* number of copies of the selected object following the same rules as **Single**. When you select **Multiple**, a dialog box for specifying the number of copies is displayed, Figure 3-81.

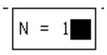


Figure 3-81. Number of Copies Dialog Box

Enter the number of copies that you want to make, and then select **Accept/Save**. After the copies have been added to the form, you must go back and rename each one so that each new object has a unique ID.

To End makes a single copy of each object, starting from the selected object to the last object on the form, following the same rules as defined for **Single.**

For example, consider that your form contains the following objects:

CS1_CM1

CS1_CM2

CS1_CM3

If you position the cursor on the line for CS1_CM2 and then choose **Edit-Copy-To End**, the result will be as follows:

CS1_CM1

CS1_CM2

CS1_CM2

CS1_CM3

CS1_CM3.

3.4.4 Application Procedures for Link Handling

This section describes link handling. A link is a relation between two objects in different structures. The following link types are supported

REFERENCES_SOLUTION	This link type indicates that a certain object belongs to a certain solution by linking the affected object and the solution's corresponding SOLUTION object together.
IS_DOCUMENTATION_OF	is used for links between document packages represented as objects in the Documentation Structure (OBJ_ID1) and objects in any structure that require documentation (OBJ_ID2).
REFERENCES_TS	This link type indicates that a certain solution represents a Typical Solution's instance by linking the corresponding SOLUTION and TYP_SOLUTION objects together. REFERENCES_TS links can not be handled manually by the user.

A link always has a direction (that is, the link is directed from OBJ_ID1 towards OBJ_ID2). A link can be either closed (connected) or open (disconnected). A link is closed when both OBJ_ID1 and OBJ_ID2 are specified. A link is open when either OBJ_ID1 or OBJ_ID2 are not specified.

Link Handling not only allows you to create links between objects, but also to update links, rename links and delete links.

3.4.4.1 Link Handling Windows

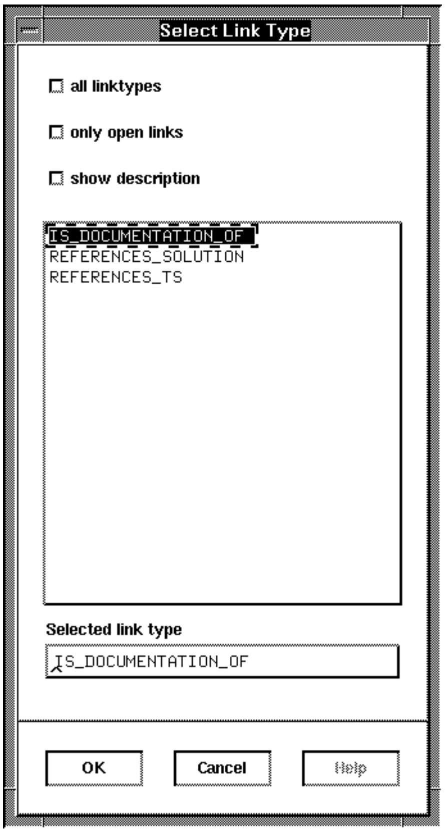
Link handling is supported by the General Link Handling window and the Link Type Specific window. The General Link Handling window provides access to links of all types for the selected object, but does not provide access to the link attributes (data). The Link Type Specific window provides access to one selected link type, and to the attributes for the selected link type.

3.4.4.2 How to Start the Link Handling Function

To start the Link Handling function:

1. Select an object then choose one of the following menu items from the **STB Link** menu: **Of Object, - Of Next Level, - Of Whole Subtree**.

   The menu item you choose determines the scope of link instances displayed in the Link Handling window. When one of these menu items is chosen, the Select Link Type dialog box is displayed, Figure 3-82.



Selected link type: IS_DOCUMENTATION_OF

Figure 3-82. Link Type Selector Box

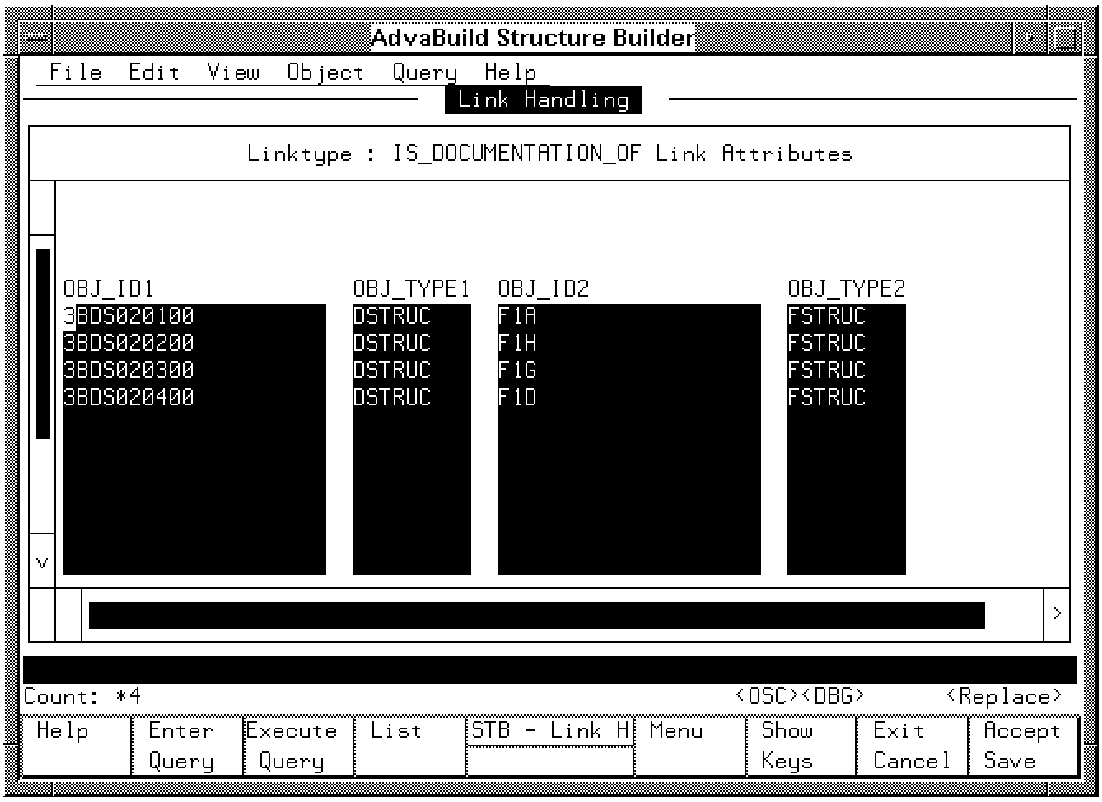
2. To open the General Link Handling window, select **all linktypes** and then click **OK**.

   To open the Link Type Specific Window first select a specific link type, and then click **OK**.

   You can further limit the scope of link instances displayed by selecting the **only open links** toggle button. The specified link instances are displayed in the appropriate window.

3.4.4.3 Link Type Specific Window

The Link Type Specific window is shown in Figure 3-83.



Linktype : IS_DOCUMENTATION_OF Link Attributes

OBJ_ID1	OBJ_TYPE1	OBJ_ID2	OBJ_TYPE2
3BDS020100	DSTRUC	F1A	FSTRUC
3BDS020200	DSTRUC	F1H	FSTRUC
3BDS020300	DSTRUC	F1G	FSTRUC
3BDS020400	DSTRUC	F1D	FSTRUC

Count: *4 <OSC><DBG> <Replace>

Figure 3-83. Link Type Specific Window

Window Organization

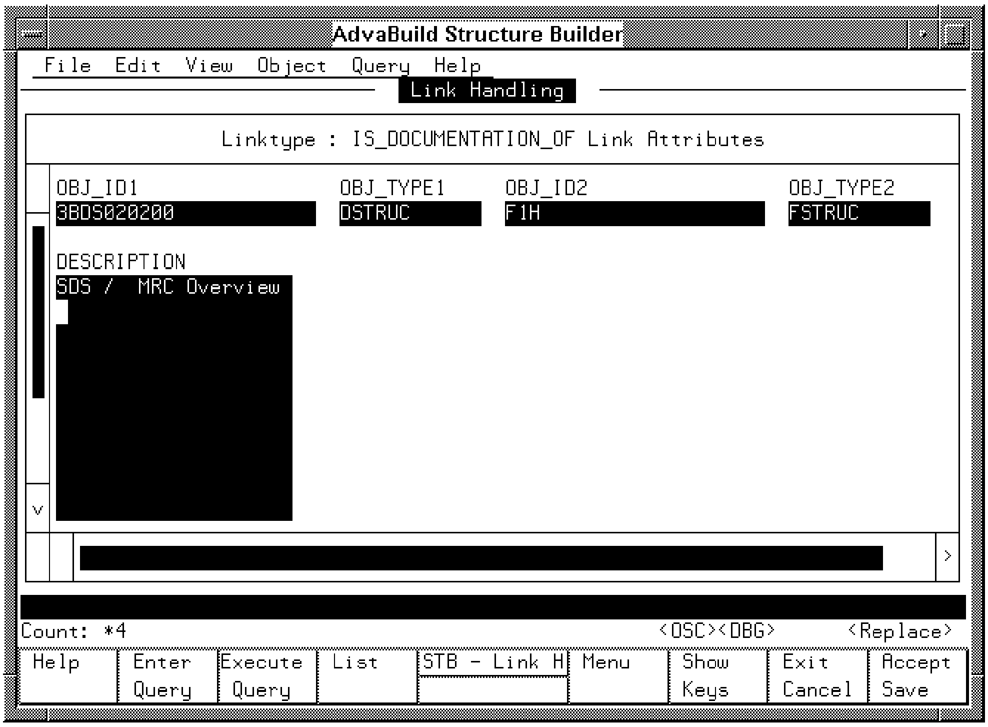
Each row in the data area represents one link. Each link has four data columns which identify the link.

- OBJ_ID1: object id for left side of link
- OBJ_TYPE1: object type for left side of link
- OBJ_ID2: object id for right side of link
- OBJ_TYPE2: object type for right side of link.

Link Attributes

Besides the link identification, a link may also have attributes. These attributes are grouped into "parts". Depending on the link type there may be several parts grouping different attributes. Each part maps into the window (exception link type **HAS_SIGNAL_ELEMENT** and **HAS_LOCATION** where the attributes are individual mapped). The selection of a part is done either via the menu or via field navigation.

For example the link type "IS_DOCUMENTATION_OF" has one additional part defined as illustrated in Figure 3-84, named 'Link Attributes'.



Linktype : IS_DOCUMENTATION_OF Link Attributes

OBJ_ID1: 3BDS020200
OBJ_TYPE1: DSTRUC
OBJ_ID2: F1H
OBJ_TYPE2: FSTRUC

DESCRIPTION: SDS / MRC Overview

Count: *4

Figure 3-84. Link Attributes Window

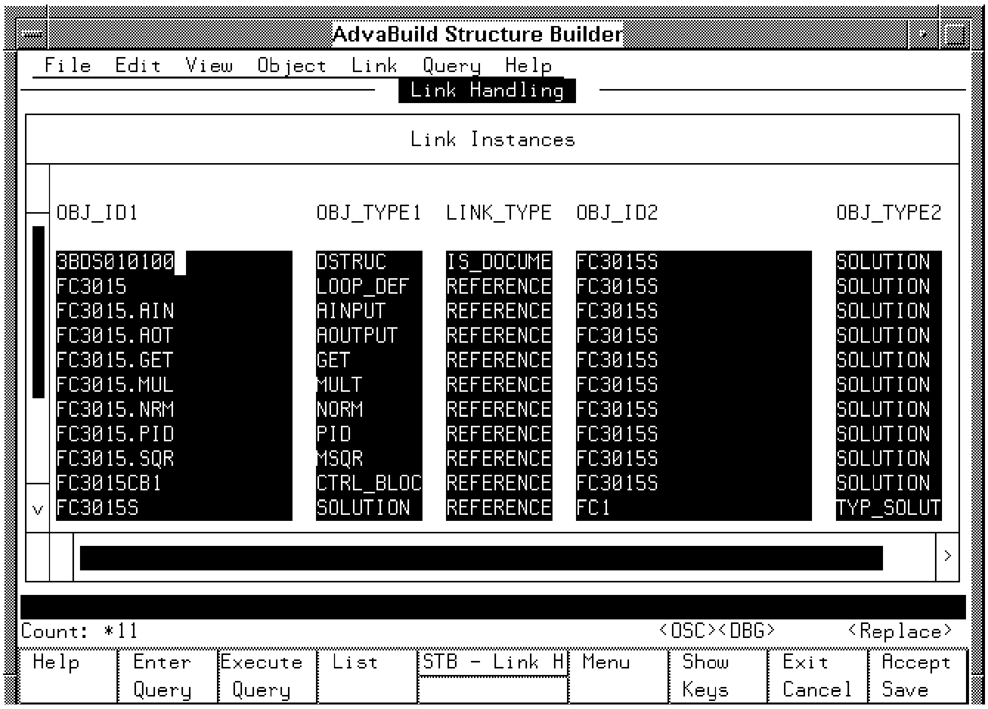
The actual link identification where the cursor is located is indicated in the top line.

3.4.4.4 General Link Handling Window

The General Link Handling Window, Figure 3-85, shows different link types, but attributes of the different links are not shown. This is the main difference for the Link Type Specific Window.

Normally you use the General Link Handling window to:

- query for links of different types
- ask for the links of a certain object that might have different links to other objects
- handle (insert, delete, query for) links that do not have attributes because these link types do not have a Link Type Specific window.



OBJ_ID1	OBJ_TYPE1	LINK_TYPE	OBJ_ID2	OBJ_TYPE2
3BDS010100	DSTRUC	IS_DOCUME	FC3015S	SOLUTION
FC3015	LOOP_DEF	REFERENCE	FC3015S	SOLUTION
FC3015.AIN	AINPUT	REFERENCE	FC3015S	SOLUTION
FC3015.AOT	AOUTPUT	REFERENCE	FC3015S	SOLUTION
FC3015.GET	GET	REFERENCE	FC3015S	SOLUTION
FC3015.MUL	MULT	REFERENCE	FC3015S	SOLUTION
FC3015.NRM	NORM	REFERENCE	FC3015S	SOLUTION
FC3015.PID	PID	REFERENCE	FC3015S	SOLUTION
FC3015.SQR	MSQR	REFERENCE	FC3015S	SOLUTION
FC3015CB1	CTRL_BLOC	REFERENCE	FC3015S	SOLUTION
FC3015S	SOLUTION	REFERENCE	FC1	TYP_SOLUT

Count: *11

Figure 3-85. General Link Handling Window

Window Organization

The window is similar to the Link Type Specific window. Each row represents one link, and each link has the following data fields:

- OBJ_ID1: object identity of the left side of the link
- OBJ_TYPE1: object type of the left side object
- LINK_TYPE: the type of link the objects are part of
- OBJ_ID2: object identity of the right side of the link
- OBJ_TYPE2: object type of the right side object.

3.4.4.5 How to Create Links

You can create links from either the Link Specific or General Link Handling window by any one of the following methods:

- manual creation of links
- copying existing links and modifying the required attributes

How to Create Links Manually

This method is typically used to create is_documentation_of link types. To create a link manually:

1. Open either the General Link Handling Window or the Link Type Specific Window.
2. Enter the object id for the left side of the link in the "OBJ_ID1" field.

   For instance, for an "is_documentation_of" link, enter the ID for the DSTRUC object.
3. Enter the object ID for the right side of the link in the "OBJ_ID2" field.

   For instance, for an "is_documentation_of" link, enter the id for the object whose documentation is being included in the documentation package represented by the DSTRUC object.
4. Enter a description (optional attribute).
5. Commit your entries by selecting **Accept/Save**.

How to Copy Links

As a document package may consist of data from different parts of the plant, there may be several links from a document package to objects in the different structures (for example, function oriented structure).

If you have to create several links for one document package you could create one link as described above and copy this link several times as follows:

1. Choose **Copy** from the **Edit** menu.

   This displays a cascade menu with three copy options: **Single**, **Multiple**, and **To End**.
2. Choose **Multiple** from the **Copy** menu.

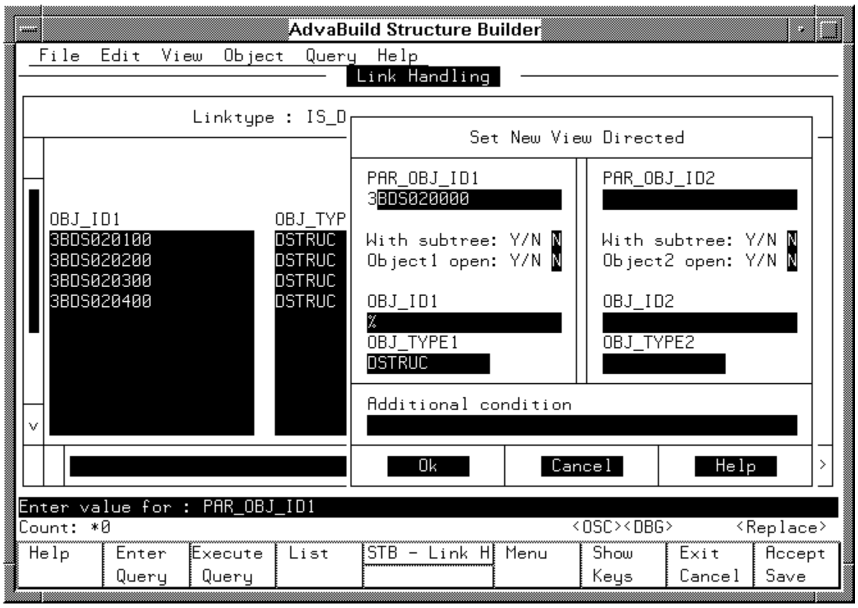
   This displays a dialog box for entering the number of copies.
3. Fill in the number of copies to be created in the dialog box.
4. Change the object identity in the "OBJ_ID2" field as required.
5. Commit your entries by selecting **Accept/Save**.

3.4.4.6 How to Query for Links

Query Using Set New View Dialog Boxes

This section provides a general description of how to query for links. When you invoke the link handling function, the Structure Builder defines a query for links based on how you open the link handling windows (General or Link Specific) as described earlier in this section.
The Structure Builder presets the values in the Set New View Directed / Undirected dialog boxes based on which object and items are selected in the Structure Builder. You can use the Set New View Directed and Set New View Undirected dialog boxes to change the view in the applicable window without having to exit the window.

To display the Set New View Directed dialog box, Figure 3-86, choose **Set Dir** from the **View** menu. There are two sets of data fields; one for the left side of the link (using suffix **1**), and one for the right side of the link (using suffix **2**). These data fields are described below.



Linktype : IS_D

OBJ_ID1	OBJ_TYP
3BDS020100	DSTRUC
3BDS020200	DSTRUC
3BDS020300	DSTRUC
3BDS020400	DSTRUC

PAR_OBJ_ID1: 3BDS020000
With subtree: Y/N N
Object1 open: Y/N N
OBJ_ID1: %
OBJ_TYPE1: DSTRUC

PAR_OBJ_ID2:
With subtree: Y/N N
Object2 open: Y/N N
OBJ_ID2:
OBJ_TYPE2:

Additional condition:

Enter value for : PAR_OBJ_ID1
Count: *0 <OSC><DBG> <Replace>

Figure 3-86. Set New View Directed Dialog Box

PAR_OBJ_ID1 (2)	These fields are for entering the object id of a parent object. This way you can limit the query to a specific subtree. For instance, retrieve all links of all boards in a specific node (where the node is identified by the parent object id).
With subtree	These fields provide the means to query for links which belong to the next level of the parent object by entering N (No) or the whole subtree of the parent object by entering Y (Yes).

Object1 (2) open	These fields provide the means to query for open links only. Specify which side of the link is open by entering **Y** (Yes) in the corresponding field. The other field must be specified as closed (**N**).
OBJ_ID1 (2)	These fields provide the means to query for all links of a specific object, or for just one specific link. To query for all links for a specific object, enter the object id in either OBJ_ID1 or OBJ_ID2 (whichever is appropriate). To query for just one specific link, specify the left side in OBJ_ID1 and the right side in OBJ_ID2.
OBJ_TYPE1 (2)	These fields provide the means to query for links involving a specific object type.

Specify query parameters in the Set New View Directed dialog box. When you are finished, commit your entries by positioning the cursor on **OK**, and then selecting **Accept/Save**.

Query using Query Menu

The **Query** menu in Link Handling windows provides the means to define a specific set of criteria to identify and retrieve one or more links from the AdvaBuild database. Refer to Section 3.4.11, Queries for details.

Note that when you use the **Query** menu only those attributes which are not already selected via the Set New View Directed dialog box (see above) can be queried. If the query is executed, the predefined values of the Set New View Directed dialog box and the values entered via the **Enter Query** menu item are merged to a joint AND-condition.

In the General Link Handling Window you can execute either the Set New View Directed dialog box or the Set New View Undirected dialog box. If the Set New View Undirected dialog box is executed last, the **Query** menu is not available.

3.4.5 How to Print Plant Documentation

There are two methods for printing documentation:

- **Print** and **PrintType** menu items from the **Object** menu in the Structure Builder menu bar

  This method is suitable for single-object documents that do not require a finished presentation
- Plant Documentation Builder.

  The Plant Documentation Builder provides an efficient method for printing project documentation as a finished deliverable documentation package. A document package contains a subset of available document types for a selected subject and application.

This section provides instructions for configuring the Plant Documentation Builder and printing documentation packages. For instructions on how to use the **Print** and **Print Type** menu items, refer to Section 3.5.5.4, Object Menu.

3.4.5.1 Plant Documentation Builder Overview

Through the Plant Documentation Builder you configure and print documentation packages. The Plant Documentation Builder generates a cover page and table of contents for each package automatically, and provides the means to label each package with revision information when you print it out. Project documentation handled by the Plant Documentation Builder includes:

- text generated via the Document Builder
- P&IDs, layout drawings, circuit diagrams generated via the Diagram Builder
- function charts generated via the Function Chart Builder
- displays generated via the Display Builder
- predefined and custom reports generated via the SQL*ReportWriter.

The Plant Documentation Builder is subdivided into two applications:

- PDB Configure Plant Documentation. Configuration activities are generally done in the start phase of an Engineering Project. Further configuration is only necessary to define working packages or to modify the contract defined applications when a project is running.

  NOTE

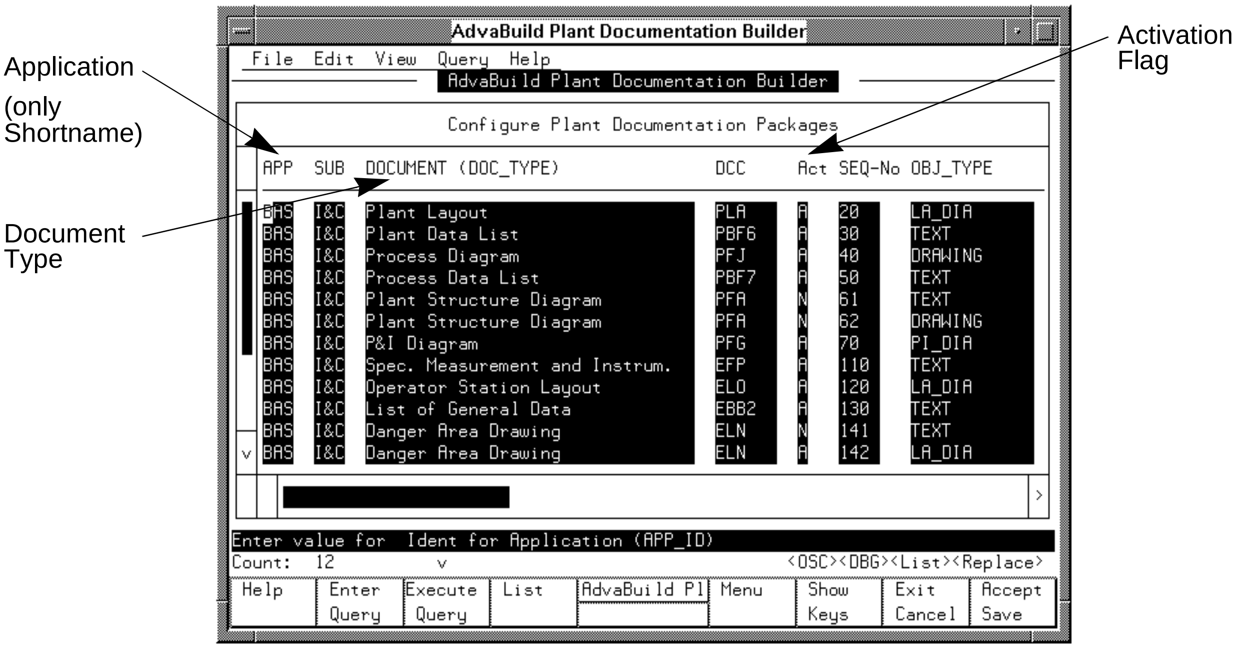
  Only the responsible person for these definitions with AdvaBuild access rights CONFIG is able to execute the corresponding configuration functions.
- PDB Generate Plant Documentation. This application is to handle the generation of Plant Documentation Packages and monitor the printout procedure for actual work-out or contract dependent documents.

To use the Plant Documentation Builder to print documentation packages, you must:

1. Configure the DOC_RELATION Table
2. Build the documentation structure and define DSTRUC object attributes
3. Create the necessary links between DSTRUC objects and objects in other structures
4. Print the documentation packages.

DOC_RELATION Table

The relations for Document Types are defined in the DOC_RELATION table in the AdvaBuild database. You can view and edit these relations via the PDB_Configure window, Figure 3-93.



APP	SUB	DOCUMENT (DOC_TYPE)	DCC	Act	SEQ-No	OBJ_TYPE
BAS	I&C	Plant Layout	PLA	A	20	LA_DIA
BAS	I&C	Plant Data List	PBF6	A	30	TEXT
BAS	I&C	Process Diagram	PFJ	A	40	DRAWING
BAS	I&C	Process Data List	PBF7	A	50	TEXT
BAS	I&C	Plant Structure Diagram	PFA	N	61	TEXT
BAS	I&C	Plant Structure Diagram	PFA	N	62	DRAWING
BAS	I&C	P&I Diagram	PFG	A	70	PI_DIA
BAS	I&C	Spec. Measurement and Instrum.	EFP	A	110	TEXT
BAS	I&C	Operator Station Layout	ELO	A	120	LA_DIA
BAS	I&C	List of General Data	EBB2	A	130	TEXT
BAS	I&C	Danger Area Drawing	ELN	N	141	TEXT
BAS	I&C	Danger Area Drawing	ELN	A	142	LA_DIA

```
Enter value for  Ident for Application (APP_ID)
Count:  12          v                    <OSC><DBG><List><Replace>
```

Figure 3-93. PDB_Configure Window

This window lists the standard document types according to their application and subject, and shows the types of documents that are active (Act). If this predefined set of documents does not meet all your requirements, you can define custom report documents via SQL*ReportWriter. The custom reports you create must be added to the DOC_RELATION table via the PDB_Configure window if you want to include the reports in documentation packages generated via the Plant Documentation Builder.

To reduce time and expense, you can limit the documents that are printed (active) for each document type according to set of deliverables as agreed upon by the customer. To do this, you selectively deactivate documents for a specific document type on an individual basis.
For instance you can deactivate LA_DIA documents for the System Functional Definition.

The Documentation Structure

A STRUC object represents the top of the documentation structure. Each documentation package is represented by a DSTRUC object under the STRUC object, Figure 3-94.

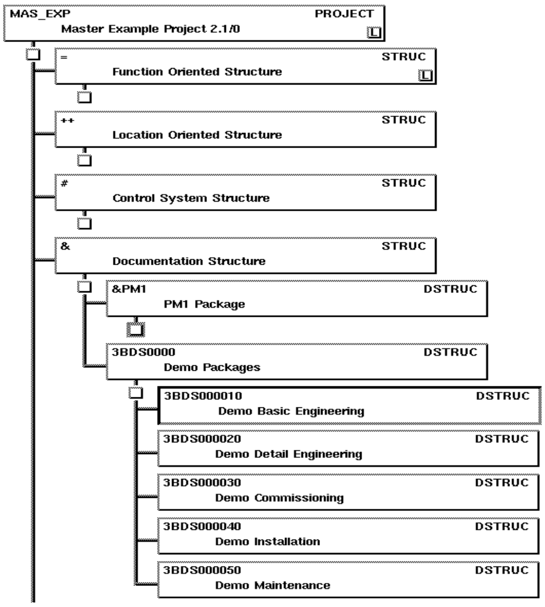


Figure 3-94. Documentation Structure

DSTRUC Object Attributes

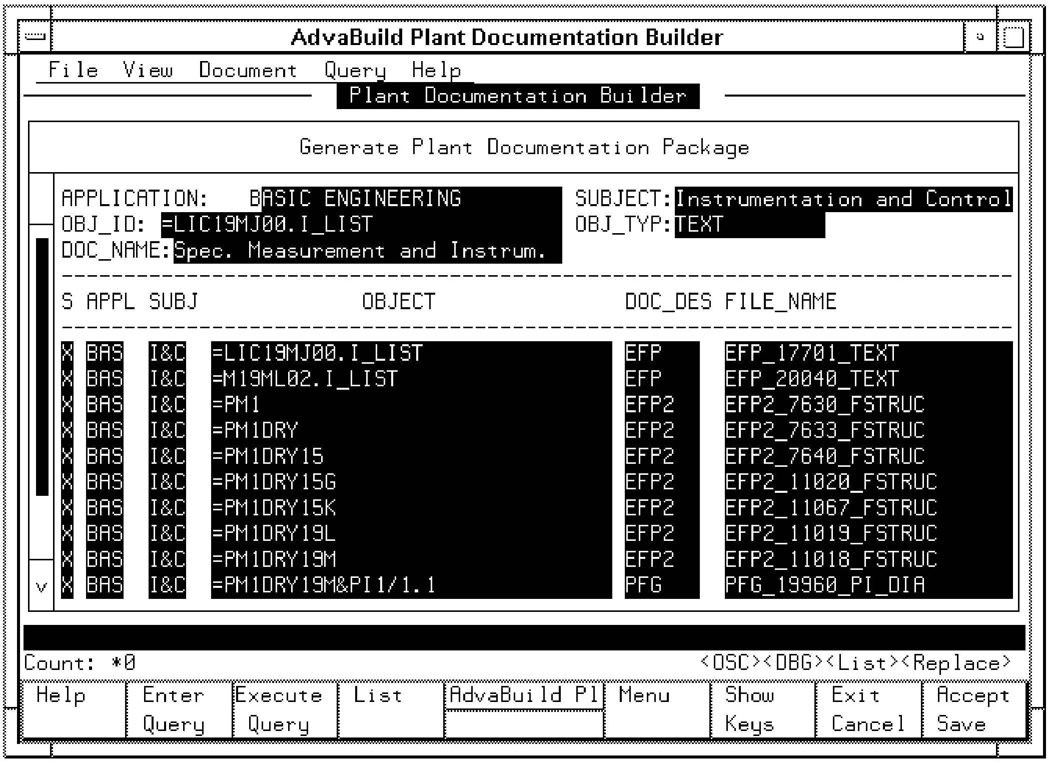
The application for the DSTRUC object is assigned via the Template Builder. Other attributes define the title and other footer information for the documentation package.

Links

IS_DOCUMENTATION_OF links can be created between DSTRUC objects and all other Object types. IS_DOCUMENTATION_OF links between DSTRUC objects and objects in the Control (#), Function (=), and Location (++) structures define the scope of the document package. When you create a IS_DOCUMENTATION_OF link (using Link menu items in the Structure Builder), all document types for the DSTRUC object's application are activated for all objects in the linked object's subtree. For instance, a link between a DSTRUC object whose application is Detail Engineering and a MP200 object, the document package represented by the DSTRUC object can print all Detail Engineering documents for all objects in this MasterPiece node structure represented by the MP200 object such as Board and Signal Reports, Function Charts, Database Lists, and so on.

Generating (Printing) Document Packages

The contents of a documentation package can be manipulated and printed via the PDB_Package_Generation window, Figure 3-95. This window lists all the documents in the document package and provides the means to print the package. Prior to printing the package, you can choose to include or exclude any document in the package on an individual basis by setting the Select (**S**) flag.



```
APPLICATION:   BASIC ENGINEERING            SUBJECT:Instrumentation and Control
OBJ_ID: =LIC19MJ00.I_LIST                   OBJ_TYP:TEXT
DOC_NAME:Spec. Measurement and Instrum.
---------------------------------------------------------------------------
S APPL SUBJ            OBJECT               DOC_DES FILE_NAME
---------------------------------------------------------------------------
X BAS  I&C  =LIC19MJ00.I_LIST               EFP     EFP_17701_TEXT
X BAS  I&C  =M19ML02.I_LIST                 EFP     EFP_20040_TEXT
X BAS  I&C  =PM1                            EFP2    EFP2_7630_FSTRUC
X BAS  I&C  =PM1DRY                         EFP2    EFP2_7633_FSTRUC
X BAS  I&C  =PM1DRY15                       EFP2    EFP2_7640_FSTRUC
X BAS  I&C  =PM1DRY15G                      EFP2    EFP2_11020_FSTRUC
X BAS  I&C  =PM1DRY15K                      EFP2    EFP2_11067_FSTRUC
X BAS  I&C  =PM1DRY19L                      EFP2    EFP2_11019_FSTRUC
X BAS  I&C  =PM1DRY19M                      EFP2    EFP2_11018_FSTRUC
X BAS  I&C  =PM1DRY19M&PI1/1.1              PFG     PFG_19960_PI_DIA

Count: *0                                       <OSC><DBG><List><Replace>
```

Figure 3-95. PDB_Package_Generation Window

Overview AdvaBuild-Structure

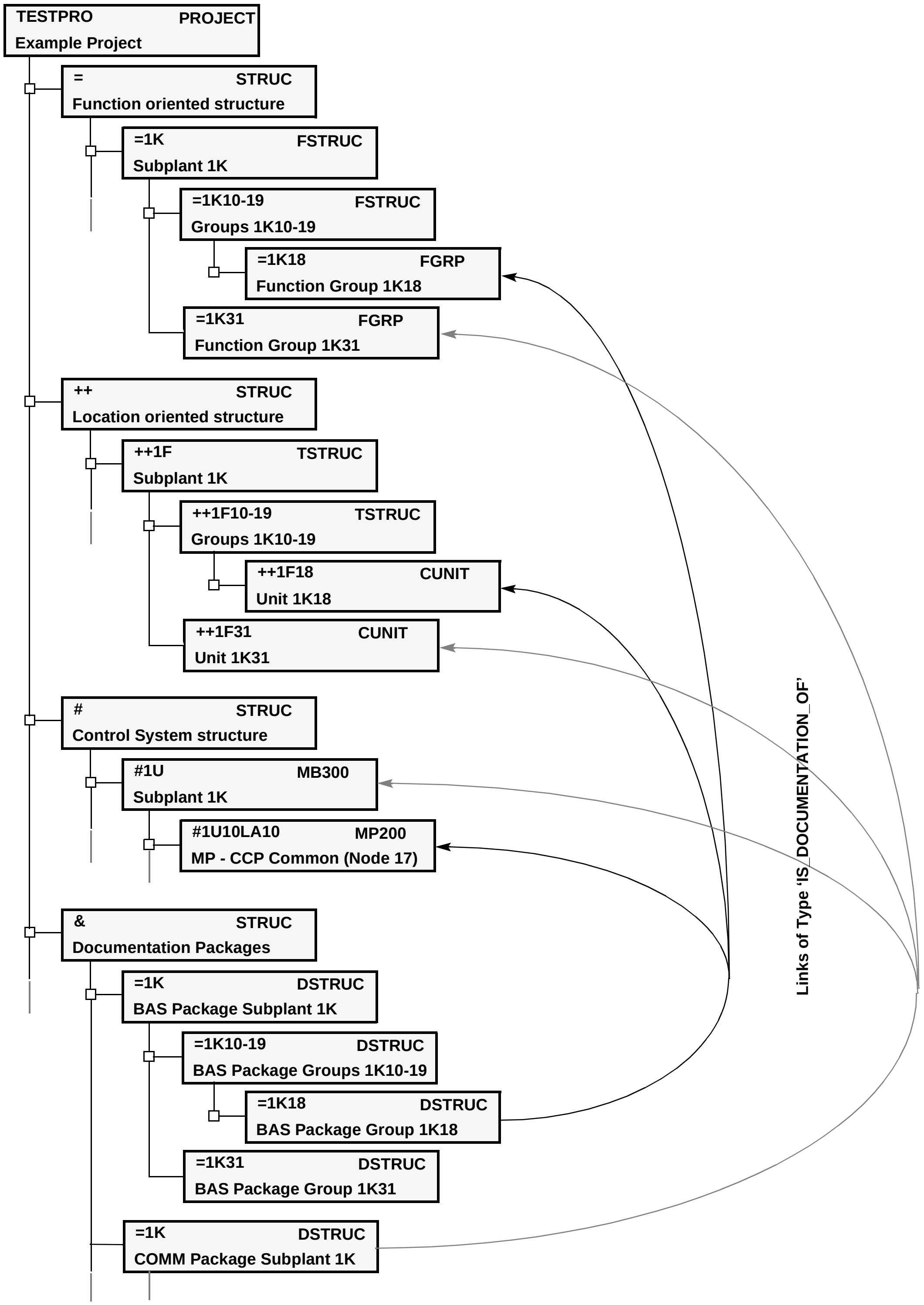


Figure 3-96. AdvaBuild-Structure with PDB Package Objects and Links

3.4.5.2 Document Types

All Document Types to be used in Plant Documentation Packages must be defined in the **DOC_RELATION** table. One document type (for example, Function Diagram, Database List and so on) can be used in many different Plant Documentation Packages if required.

If a document type is not included in this table, then a document of that type specified in a Plant Documentation Package, can not be printed out. The standard-document-packages for a complete project are predefined in the **DOC_RELATION** table.

For List of predefined Document Types see chapter 'List of DocumentTypes for Master', ,

or use the STB menu item **Object Prin**t started from a **DSTRUC object.**

- View **DOC_TYPES** for the list of all Document Types and
- View **DOC_STD** for the list of predefined Plant Documentation Package Standard

Table 3-6. Active DocumentTypes for Master

DOCUMENTS (Document Type)	**Builder responsible**	**Obj_Type (connected)**	**Query Table(s)**	**View**
List of Designations	DOB	TEXT		DOCUMENT
Plant Layout	DIB	LA_DIA		DIAGRAM
Plant Data List	DOB	TEXT		DOCUMENT
Process Diagram	DIB	DRAWING		DIAGRAM
Process Data List	DOB	TEXT		DOCUMENT
P&I Diagram	DIB	PI_DIA		DIAGRAM
Control Structure Diagram	DOB, DIB	TEXT, DRAWING		DOCUMENT DIAGRAM
Function Diagram (Basic)	DOB, DIB	TEXT, DRAWING		DOCUMENT DIAGRAM
Function Description (Basic)	DOB	TEXT		DOCUMENT
Spec. Measurement and Instrum.	DOB	TEXT		DOCUMENT
Operator Station Layout	DIB	LA_DIA		DIAGRAM
List of General Data	DOB	TEXT		DOCUMENT
Danger Area Drawing	DOB, DIB	TEXT, LA_DIA,		DOCUMENT DIAGRAM
Instrumentation List	ReportWriter	FSTRUC, FGRP, TAG	DEV_FUN DB-Elements in TAG	INSTRUM_LIST

Table 3-6. Active DocumentTypes for Master

DOCUMENTS (Document Type)	**Builder responsible**	**Obj_Type (connected)**	**Query Table(s)**	**View**
Overview Diagram Control Syst.	DOB, DIB	TEXT, DRAWING		DOCUMENT DIAGRAM
Basic Cable Routing Drawing	DIB	DRAWING		DIAGRAM
Overview Arrang.+ Interconn.	DIB	DRAWING		DIAGRAM
Circuit Diagram Input/Output	DIB	CI_DIA		DIAGRAM
Assembly Drawing	DIB	DRAWING HOOKUP		DIAGRAM DIAGRAM
Drawing Arrangements and Interconnections	DIB	DRAWING		DIAGRAM
Arr.Draw.Items in Constr.Units	DIB	LA_DIA		DIAGRAM
Circuit/Interconnection Diagr.	DIB	CI_DIA		DIAGRAM
Arr. and Interconn. Drawing	DIB	DRAWING		
Item Location Diagram	DIB	LA_DIA		
Function Diagram (Detail)	DOB, DIB	TEXT, DRAWING		DOCUMENT DIAGRAM
Function Description Type 01	DOB	TEXT		DOCUMENT
Database List	FCB	MP200,		DB_LIST
Function Diagram	FCB	MP200,		FUNCT_CHART
Process Display	DYB	DISP		DISPLAY
Equipment List		FSTRUC		DEVICE_LIST
Spare Part List		TSTRUC		SPARE_PARTS
Text Description (FrameMaker)	DOB	TEXT		DOCUMENT
Free Drawing (CAD-Tool)	DIB	DRAWING		DIAGRAM
Device List	ReportWriter -Interface	TSTRUC, RACK CUNIT SUBUNIT	DCS_DEV	DEVICE_LIST

Table 3-6. Active DocumentTypes for Master

DOCUMENTS (Document Type)	**Builder responsible**	**Obj_Type (connected)**	**Query Table(s)**	**View**
Signal List Input/Output	ReportWriter - Interface	PROJECT, FSTRUC, FGRP, TAG, MB300 MB90, MP200 AC110C AC410C AC450C AC70C	all Signal-Tables	SIGNAL_LIST
Device List of Master-Boards	ReportWriter - Interface	PROJECT, FSTRUC, FGRP, TAG, MB300 MB90, MP200 AC110C AC410C AC450C AC70C	all Boards for Master	BOARD_LIST
I/O Channel - Signal Assignment (Board-channels with and without signals - started with board-objects)	ReportWriter - Interface	MB300 MB90, MP200 AC110C AC410C AC450C AC70C	all Boards and Signals	IO_LIST
Connected Signals (IO_CONNECTED) (Board-channels only with connected signals - started with board-objects)	ReportWriter - Interface	MB300 MB90, MP200 AC110C AC410C AC450C AC70C	all Boards and Signals	IO_CONNECTED
Connected Signals (HSELINKS) (Board-channels only with connected signals - started with signal-objects)	ReportWriter - Interface	FSTRUC FGRP	all Boards and Signals	HSELINKS

Table 3-6. Active DocumentTypes for Master

DOCUMENTS (Document Type)	**Builder responsible**	**Obj_Type (connected)**	**Query Table(s)**	**View**
Connected PC-Programs (HCTLINKS) (PC-Programss only with connected signals - started with signal-objects)	ReportWriter - Interface	PCPGM	all PC-Programs and Signals	HCTLINKS
Signal List AC400	ReportWriter - Interface	AC410C AC450C	All AC400 signals	AC400_SIGS
Device List of AC400 Boards	ReportWriter - Interface	AC410C AC450C	All AC400 Boards	AC400_BOARDS
Signal List AC110	ReportWriter - Interface	AC110C MB90 TAG	All AC110 signals	AC110_SIGS
Device List of AC110 Boards	ReportWriter - Interface	AC110C MB90 TAG	All AC110 Boards	AC110_BOARDS
Signal List AC70	ReportWriter - Interface	AC70C MFB	All AC70 signals	AC70_SIGS
Device List of AC70 Boards	ReportWriter - Interface	AC70C MFB	All AC70 Boards	AC70_BOARDS
Device List of MP200 Boards	ReportWriter - Interface	MP200 MB300	All MP200 Boards	MP200_BOARDS
Signal List MP200	ReportWriter - Interface	MP200	All MP2070 signals	MP200_SIGS
Instrumentation List	ReportWriter - Interface	FGRP FSTRUC TAG		
List of Measurement Points	ReportWriter - Interface	FGRP FSTRUC TAG		
List of DCS-Devices	ReportWriter - Interface	CUNIT TSTRUC RACK SUBUNIT	DCS_DEV	

3.4.5.3 How to Configure the DOC_RELATION Table

This is a set-up activity that is generally done before you build documentation packages. This activity can only be performed by a user with access at the CONFIG level or higher, and is only required in two situations:

- when you want to turn off certain document types for a specific application
- when you need to add a custom report to the DOC_RELATION table.

Access to the Documentation_Relation table is provided via the PDB_Configure window, Figure 3-93. To display this window, choose **Configure Plant Documentation Builder** from the **Utility** menu in the Structure Builder menu bar.

How to Turn Off Document Types for Certain Applications

Each record in the DOC_RELATION Table has an Activation flag (Act) that determines whether or not the corresponding document is included in the documentation package for the application. The default state for the flag is **A** (Active). This means the document is included when the package is printed. To deactivate the document, you can set the activation flag to either **H** (Hibernate or temporarily suppressed) or **N** (Not Active). To set this flag:

1. Scroll through the list of documents using the up/down arrow keys. When you find a document for a specific application and document type that you DO NOT want to include in the package, use <**Tab**> to navigate to the Act column, and enter **N** (not active).
2. Repeat step 1 for as many document views as required.
3. Press **Accept/Save** when you are done.

How to Add a New Document Type to the DOC_RELATION Table

To apply the Plant Documentation Builder functions to a custom report you created via the SQL*ReportWriter or another new Document Type, the document or report must be entered in the DOC_RELATION table. To do so:

1. Position the cursor on the record above where you want to insert a new record.
2. Press **Menu** (<**f5**>) to jump into the PDB_Configure window menu bar.
3. Choose **Insert** from the **Edit** menu. This inserts an empty record under the record you selected in step 1.
4. Fill in the fields of the empty record as described below. Use <**tab**> and <Shift-**tab**> to move forward and backward between fields. Press **Accept/Save** when you are done.
5. If a new Document Type is created, related to objects of type
   TEXT, DRAWING, HOOKUP, LA_DIA, PI_DIA, CI_DIA, an additional entry is inserted into table OBJ_LIST_OF_VALUES for the corresponding Object Type.
   The LOV insert procedure will be logged into a file "**lov_<no>_<obj_type>_ins.log**"
6. If the last entry of a Document Type (in table DOC_RELATION)
   during configuration will be deleted, and the DOC_TYPE is not used
   by dependent objects, then the corresponding entry in the table
   OBJ_LIST_OF_VALUES will be deleted.
   The LOV delete procedure will be logged into a file **"lov_<no>_<obj_type>_del.log**"
   <no> is a running number during one Configure session.

For new created reports, the corresponding Print Method will be added automatically into the system. All other Document Types (DOB, DYB, DIB, FCB) can be related only to Print Methods existing in the STB

How to Add a New Application / Subject to the DOC_RELATION Table

To extend the Plant Documentation Builder functions for new Subjects / Applications:

1. Define the Subject / Application and compose the set of Document Types desired.
2. In the PDB Configure menu enter the new entries by edit or copy/modify.
3. Creation of a new SUBJECT or APPLICATION during PDB - Configuration inserts an additional entry into table OBJ_LIST_OF_VALUES for DSTRUC-objects, attribute SUBJECT or APPLICATION and can be used in future for DSTRUC-objects.
   The LOV insert procedure will be logged into a file "**lov_<no>_<attribute>_ins.log**"
4. If the last entry of an APPLICATION or SUBJECT (in table DOC_RELATION) during configuration will be deleted, and the APPLICATION or SUBJECT is not used by DSTRUC objects, then the corresponding entry in the table OBJ_LIST_OF_VALUES will be deleted and the APPLICATION or SUBJECT cannot be used in future.
   The LOV delete procedure will be logged into a file "**lov_<no>_<attribute>_del.log**"
   <no> is a running number during one Configure session.

Attributes of table DOC_RELATION

The attributes **APP_ID** and **SUB_ID** are only display attributes as abbreviations of APPLICATION and SUBJECT (better overview on different forms and reports)

• **APPLICATION**	**APP_ID**	
– BASIC ENGINEERING	BAS	(predefined)
– COMMISSIONING	COM	(predefined)
– DETAILED ENGINEERING	DET	(predefined)
– INSTALLATIONINS	INS	(predefined)
– MAINTENANCE	MNT	(predefined)
– PLANT OPERATION	PLA	(predefined)

• **SUBJECT**	**SUB_ID**	
– Instrumentation and Control	I&C	(predefined)
– Energy Supply	ESP	
– Air Conditioning	AIR	
– Production/Operating Control	POC	
– Data Acquisition	DAC	

- **DOC_TYPE**

  For List of predefined Document Types see Table 3-6 or activate the correspondent Report DOC_STD.

- **TYP_ID**

  The Document Kind Classification Code (DCC) according to the Standard IEC-SC3B/WG4.

  The DCC is part of the spoolfile name and **has to be unique.** Otherwise the spoolfiles can be overwritten.

  Recommendation:

  The last position of predefined DCC represents the correspondent ObjectType can be organized according the following keys:

Key	ObjectType
1	PROJECT (1)
1	FSTRUC (1)
1	FGRP (1)
2	TAG
3	MB90
4	MB300
5	MP200
6	AC110
7	AC410
8	AC450
9	AC70
0	MFB

(1) The usage of only one ObjectType PROJECT or FSTRUC or FGRP is useful.

- **PCK_SEQ_NO**

  Sequence number of the Document Type in the actual Package Range of Document in Printout Sequence embedded in the CAPE Hierarchy (ordered by Sequence number, corresponding object or vice versa)

  Recommendation:

  The last position of predefined sequence Number should be representing the correspondent ObjectType. For description see TYP_ID (DCC).

- **OBJ_TYPE**

  Object_Type for definition of Document <-> Object Relation

- **COMMAND**

  Method (Command) to activate Printout.

- **COM_VIEW**

  Additional Description of Command

- **ID_ATTR**

  An additional ID-Attribute exist for Subselect of Object_Type (Relation 'Document and Attribute' to Obj_type). The ID_ATTR is used for TEXT-, Display- and Drawing-Objects.

Additional Attribute exists	'Y'
No additional Attribute	'N'

- **STANDARD**

  Not supported

- **QUEUE TYPE**

Queue Type	Description	Environment
AS	ASCII Queue	C_PRINT_ASCII
AD	ASCII duplex Queue	C_PRINT_ASCII_D
PS	PostScript Queue	C_PRINT_POST
PD	PostScript duplex Queue	C_PRINT_POST_D
GL	HP-GL Queue	C_PRINT_HPGL
GD	HP-GL duplex Queue	C_PRINT_HPGL_D
AX	XL300 ASCII Queue	C_PRINT_ASCII_XL
PX	XL300 PostScript Queue	C_PRINT_POST_XL
GX	XL300 HP-GL Queue	C_PRINT_HPGL_XL
LR	Remote Queue	C_PRINT_LPRAW
DX	Display Builder Queue	DYB_PRINTER

- **FOOTER**

  Not supported

- **F_INSERT**

  Instructions for Package Sort (single choice): Include the Document Type [X]

- **F_REPLACE**

  Instructions for Package Sort (single choice): Replace the Document Type [X]

- **F_REMOVE**

  Instructions for Package Sort (single choice): Remove the Document Type [X]

- **PRINT_FORMAT**
- **SUBTREE_HNDL**

  Specification of Subtree-Handling

Handle DocumentType	in Subtree	Inclusive	'I'
		Exclusive	'E'
	no Subtree - Handling		'N'

- **ACTIVATION**
- Document will be generated / temporary suppressed / not in Package

Document will be generated	(Activ)	'A'
temporary suppressed	(Hibernate)	'H'
not in Package	(Not activ)	'N'

- **W_TIME**

  Waiting Time in seconds (1 - 999) for Delivery of Spoolfile by called Tool

- **CYCLES**

  Number of Waiting cycles (1 - 999)
  (Wait CYCLES times 'W_TIME' seconds for Print Order results).

3.4.5.4 How to Build the Documentation Structure

To establish the documentation structure, first insert a STRUC object to represent the top of the Document structure subtree in the project. The STRUC object is a child of the PROJECT object. To insert the STRUC object under the PROJECT object:

1. Click on the Project object in the project tree presentation.
2. Choose **Insert** from the **Object** menu.

   When you insert an object under the Project object, the Insert Object dialog box provides a number of choices for object type.
3. Select **STRUC**.
4. Enter a suitable object ID and comment to identify this object as the top of the Document structure (for example: **&** for object ID, and **Document Packages** for a comment).
5. When you are finished with the Insert Object dialog, click on **OK**.

Now insert DSTRUC objects as required under the STRUC object. Each DSTRUC object represents a different documentation package. To insert the DSTRUC objects:

1. Click on the STRUC object you just inserted.
2. Choose **Insert** from the **Object** menu.

   When you insert an object under the STRUC object, the Insert Object dialog box provides a number of choices for object type.
3. Select **DSTRUC**.
4. Enter a suitable object ID and comment to identify the specific document package.
5. When you are finished with the Insert Object dialog, click on **OK**.
6. Repeat steps 1 through 5 to insert additional DSTRUC objects as required.

3.4.5.5 How to Define the Object Attributes for a Documentation Package

Open the Template Builder view of that object to define the object attributes. One attribute is the Application. This attribute determines the basic contents of the documentation package as indicated in Table 3-6. The other attributes define the title and other information provided on the cover sheet for the documentation package.

NOTE

You must specify the DOCUMENT TYPE attribute for TEXT, LA_DIA, DRAWING, PI_DIA, and CI_DIA objects that you intend to link to DSTRUC objects. This is because each instance of these object types can only be associated with one specific document type. If you don't specify the document type, document view for the object will not be included in the document package.

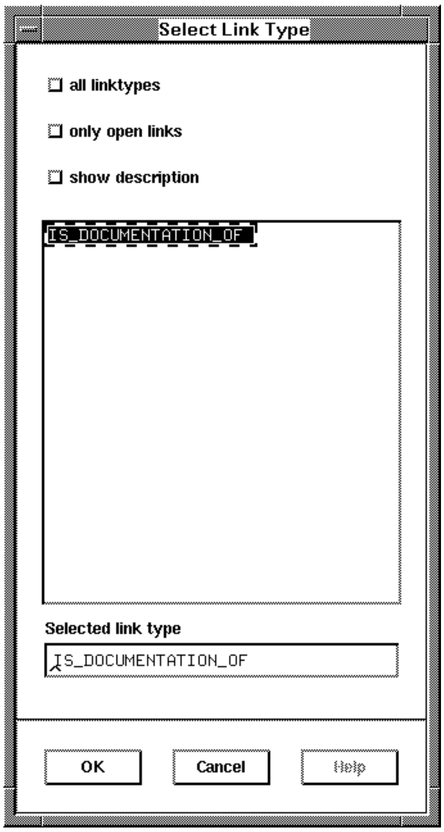
3.4.5.6 How to Create Links for DSTRUC Objects

You define the scope of the documentation package by defining IS_DOCUMENTATION_OF links between the representative DSTRUC object and applicable objects in the Control, Function, and Location structures. When you link a DSTRUC object to an object in another structure, all of the descendant objects are also linked to the DSTRUC object.

To define an IS_DOCUMENATION_OF link:

1. In the Structure Builder menu bar, choose **Open Type** from the **Link** menu.

   This displays the Link Types selector box, Figure 3-97.

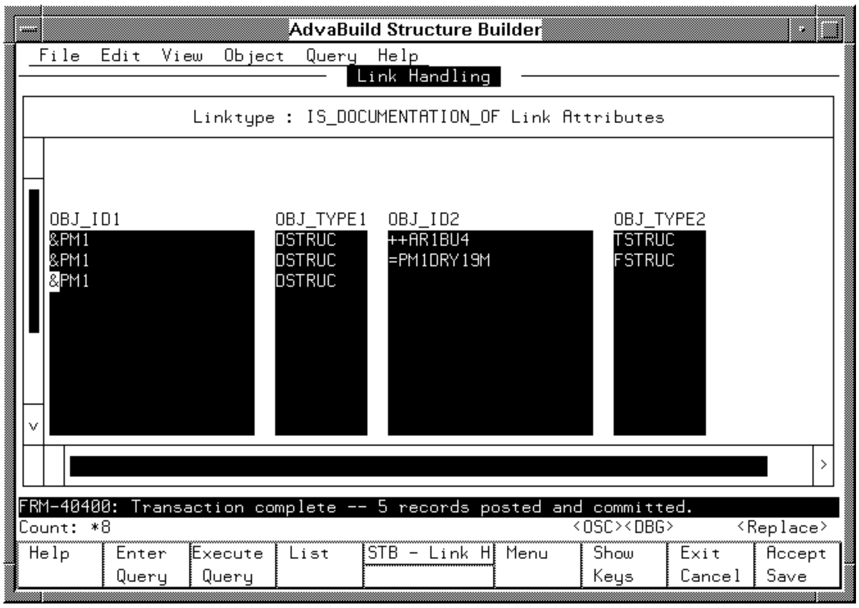


Selected link type: IS_DOCUMENTATION_OF

Figure 3-97. Link Type Selector Box

2. Click on **IS_DOCUMENTATION_OF** link type.
3. Press <**Return**> or click on **OK** to select it.

This opens the Link Handling window for the selected link type, Figure 3-98.



Linktype : IS_DOCUMENTATION_OF Link Attributes

OBJ_ID1	OBJ_TYPE1	OBJ_ID2	OBJ_TYPE2
&PM1	DSTRUC	++AR1BU4	TSTRUC
&PM1	DSTRUC	=PM1DRY19M	FSTRUC
&PM1	DSTRUC		

```
FRM-40400: Transaction complete -- 5 records posted and committed.
Count: *8                                  <OSC><DBG>      <Replace>
```

Figure 3-98. Link Handling Window

4. Enter the object ID for the DSTRUC object in the "OBJ_ID1" field.
5. Enter the object ID for the right side of the link in the "OBJ_ID2" field.
6. Enter a description (optional attribute).
7. Commit your entries by selecting **Accept/Save**.

3.4.5.7 How to Generate PDB Packages

1. Design the Document Structure and the content of different Packages according to Application and Plant-Structures

   Example:

   – Applications BASIC ENGINEERING
   COMMISSIONING

   – for Subplant 1P11
   1P18
   1P31

2. Generate the corresponding Package-Objects (Type 'DSTRUC') in the **AdvaBuild** Structure. (**Structure Builder**)

   Example:

   – Object named 3BDS070211 for Package BASIC ENGINEERING of Subplant 1P11

   – Object named 3BDS070218 for Package BASIC ENGINEERING of Subplant 1P18

- Object named 3BDS070231 for Package BASIC ENGINEERING of Subplant 1P31
- Object named 3BDS080211 for Package COMMISSIONING of Subplant 1P11
- Object named 3BDS080218 for Package COMMISSIONING of Subplant 1P18
- Object named 3BDS080231 for Package COMMISSIONING of Subplant 1P31
- Fill in the (predefined) Links to the (Structure-) Objects in the **AdvaBuild** -Structures (Function oriented -, Location oriented - and Control System structure) to define the scope of the Plant Documentation Packages.

Example:

Link of Type 'IS_DOCUMENTATION_OF' from Object 3BDS080231 to Object '=PM1DRYF1A'

Link of Type 'IS_DOCUMENTATION_OF' from Object 3BDS080231 to Object '++AR1L1A'

Link of Type 'IS_DOCUMENTATION_OF' from Object 3BDS080231 to Object '#N11AREA1'

and so on

3. Use the PDB_Configure window to modify the Content of Plant Documentation Packages for each Application.

   Example:

   Reduce the number of Document Types contained in the desired Packages

4. Use the PDB_Package_Generation to start the printout of Plant Documentation Packages
   - Start with a 'DSTRUC'-Object
   - Modify the Order List (Reduce the number of Print Orders).
   - Start the Printout (menu item **Document Print** of **PDB_Package_Generation**)

3.4.5.8 How to Print Document Packages

You print document packages via the PDB_Generation window, Figure 3-99. This window provides the means turn off (not print) certain documents in the package on an individual basis prior to printing the package.

How to Access the PDB_Package_Generation Window

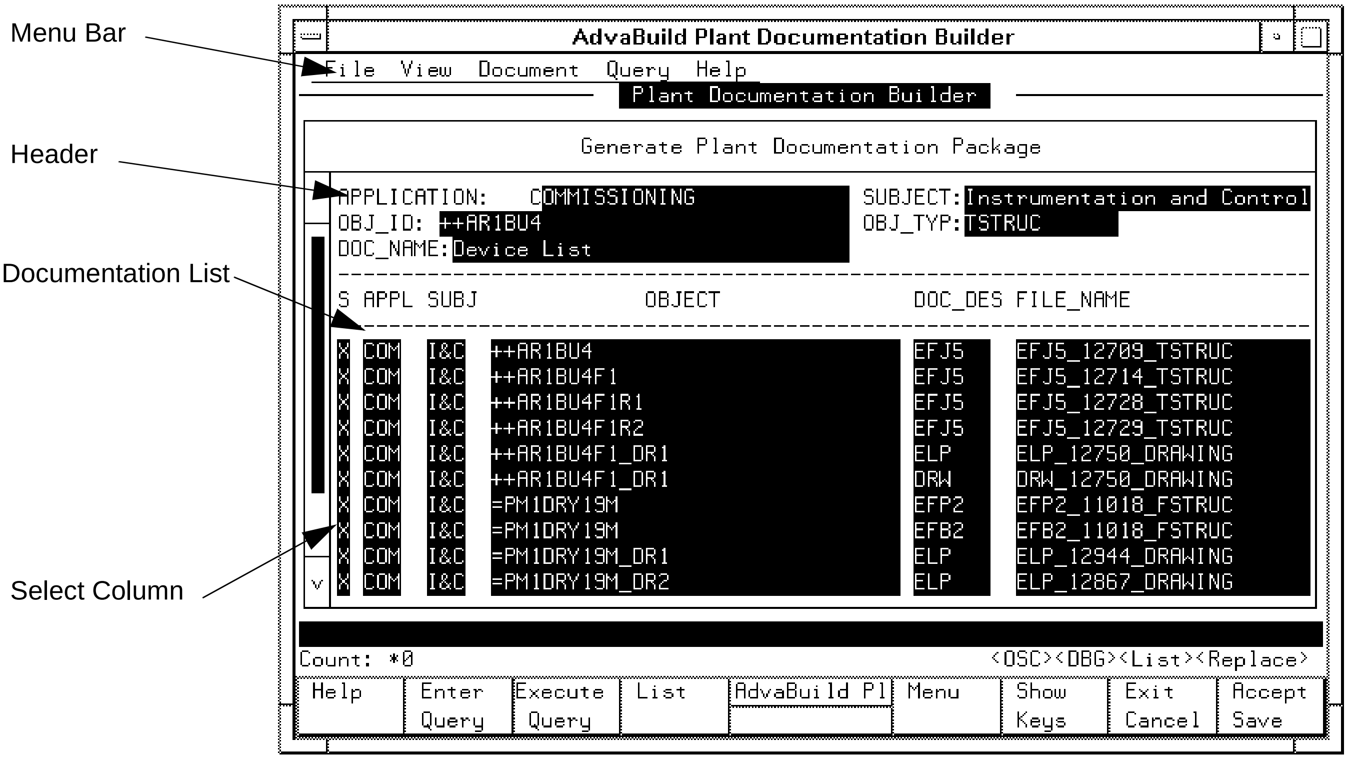
To display the PDB_Package_Generation window:

1. Select the object that you want to generate the documentation package for.

   The operation of the PDB_Package_Generation window varies depending upon whether you access it from a DSTRUC object or an object in another structure. You should select a DSTRUC object when you intend to generate a finished deliverable documentation package. You can select an object in another structure for preliminary and/or partial printouts.

2. Choose **Plant Documentation Package** from the **Utility** menu in the Structure Builder menu bar.

   This displays the PDB_Package_Generation Window. This window has three parts: the menu bar, header part, and document list.



```
                 Generate Plant Documentation Package

APPLICATION:   COMMISSIONING             SUBJECT:Instrumentation and Control
OBJ_ID: ++AR1BU4                         OBJ_TYP:TSTRUC
DOC_NAME:Device List
-----------------------------------------------------------------------------
S APPL SUBJ           OBJECT                DOC_DES FILE_NAME
-----------------------------------------------------------------------------
X COM  I&C  ++AR1BU4                        EFJ5    EFJ5_12709_TSTRUC
X COM  I&C  ++AR1BU4F1                      EFJ5    EFJ5_12714_TSTRUC
X COM  I&C  ++AR1BU4F1R1                    EFJ5    EFJ5_12728_TSTRUC
X COM  I&C  ++AR1BU4F1R2                    EFJ5    EFJ5_12729_TSTRUC
X COM  I&C  ++AR1BU4F1_DR1                  ELP     ELP_12750_DRAWING
X COM  I&C  ++AR1BU4F1_DR1                  DRW     DRW_12750_DRAWING
X COM  I&C  =PM1DRY19M                      EFP2    EFP2_11018_FSTRUC
X COM  I&C  =PM1DRY19M                      EFB2    EFB2_11018_FSTRUC
X COM  I&C  =PM1DRY19M_DR1                  ELP     ELP_12944_DRAWING
X COM  I&C  =PM1DRY19M_DR2                  ELP     ELP_12867_DRAWING

Count: *0                                         <OSC><DBG><List><Replace>
```

Figure 3-99. PDB_Package_Generation Window

The header part contains the following fields:

APPLICATION	This is the selected application for the documentation package. When you access this window from a DSTRUC object, the application is determined by the APPLICATION attribute defined for the DSTRUC object and cannot be changed. When you access this window from another object, you can select the application that you want to generate a package for. The choices are: **BASIC ENGINEERING**, **DETAIL ENGINEERING**, **INSTALLATION**, **COMMISSIONING**, **PLANT OPERATION**, and **MAINTENANCE**.
SUBJECT	This is the subject of the application. The only valid subject is: **Instrumentation & Control**.
OBJ_ID	This is the object ID of the object related to the document that the cursor is currently on in the document list part of the window. The entry in this field changes as you scroll through the list.

OBJ_TYP	This is the object type of the object related to the document that the cursor is currently on in the document list part of the window. The entry in this field changes as you scroll through the list.
DOC_NAME	This is the Document Type of the document that the cursor is currently on in the document list part of the window. The entry in this field changes as you scroll through the list.

The document list part shows the individual documents in the selected documentation package. When you access this window from a DSTRUC object, the list contains all documents for the application assigned to that object. When you access this window from another object, the documents shown depend upon the application that you select in the header part. Each document is represented by a row with the following columns:

S	This is the select flag for the document. The default status is active (indicated by **X**). When the select flag is active, the document is included when the document package is printed. If you do not want to include the document, delete the **X** by pressing the spacebar.
APPL	This is the defined abbreviation of application as specified in the APPLICATION field above.
SUBJ	This is the defined abbreviation of subject of the application as indicated in the SUBJECT field above.
OBJECT	This is the ID for the object that the document is associated with.
DOC_DES	This is the document type for the document.
FILE_NAME	This is the filename of the output file without the AdvaBuild extension (.ps). The filename is created using the following conventions: <DOC_DES>_<INT_KEY>_<DES_TYPE> where INT_KEY is the internal key attribute of the corresponding object.

How to Navigate Between Header and Document List

To move the cursor between the header and document list, use the <**Next**> and <**Prev**> keys.

How to Turn Off Selected Documents in the Documentation Package

To turn off a document:

1. Move the cursor into the document list part of the window by pressing <**Next**>.
2. Use the up or down arrow keys to scroll through the document list.
3. Select or deselect the individual documents by setting the associated select (S) column (**X** = active, blank = not active).

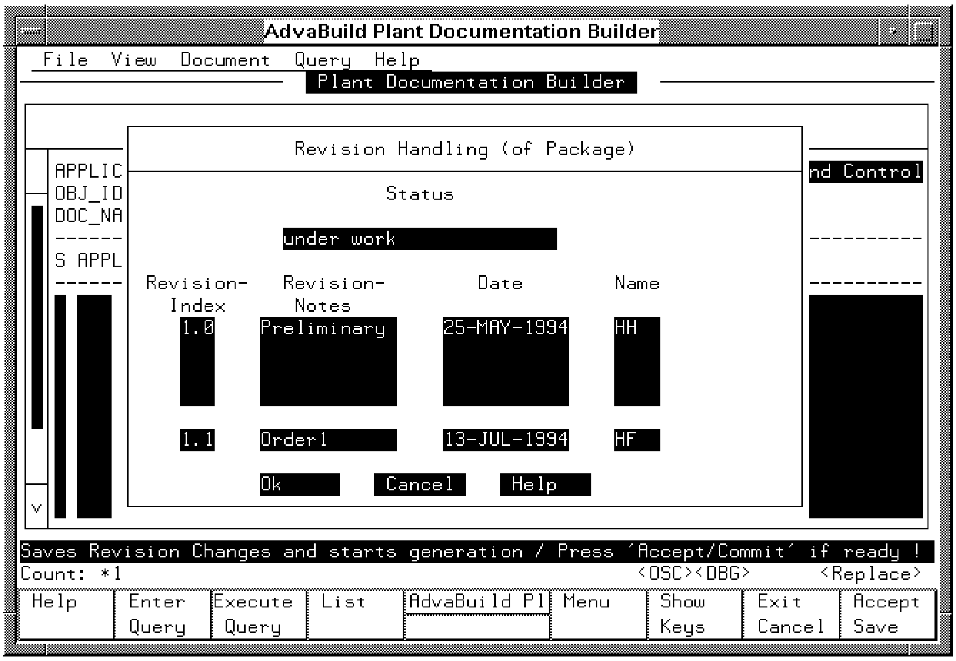
When you want to print just a few documents, you can turn all the documents off by choosing **Deselect All** from the **Document** menu. This way you can select just the objects that you want to print.

How to Print a Documentation Package

To print a documentation package:

1. Choose **Print** from the **Document** menu in the PDB_Package_Generation menu bar.

   This displays the Revision Handling dialog box, Figure 3-100. If this documentation package has been printed before, and revision handling information has been entered, that information is displayed in this dialog box. If not, then all the fields are blank.



Revision Handling (of Package)

Status: under work

Revision-Index	Revision-Notes	Date	Name
1.0	Preliminary	25-MAY-1994	HH
1.1	Order1	13-JUL-1994	HF

```
Saves Revision Changes and starts generation / Press ´Accept/Commit´ if ready !
Count: *1                                        <OSC><DBG>      <Replace>
```

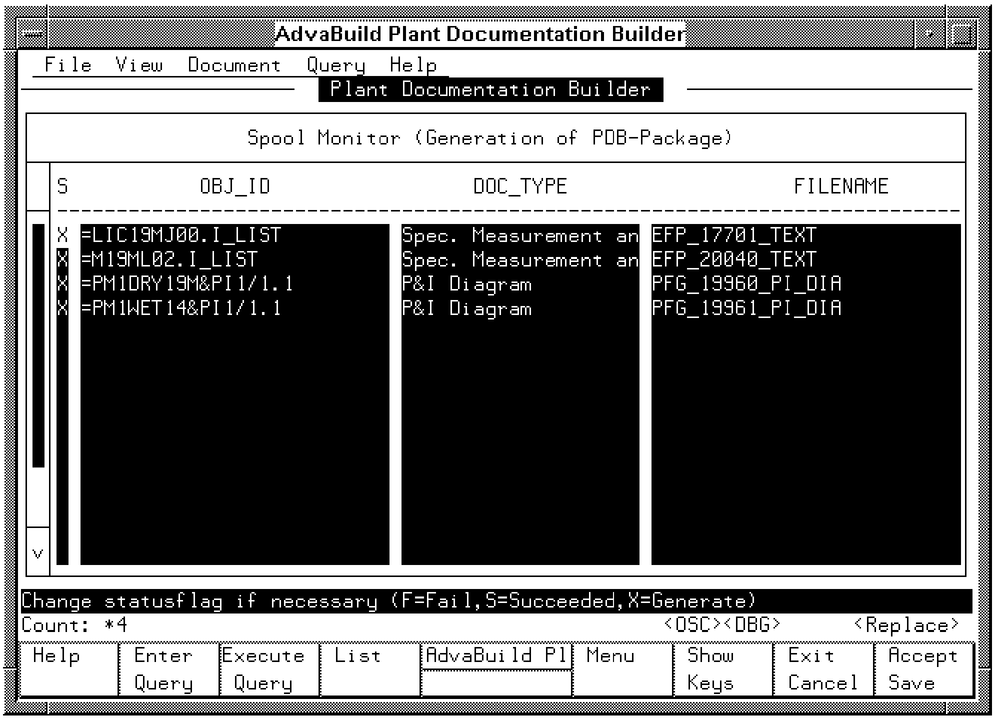
Figure 3-100. Revision Handling Dialog Box

2. Update the revision information as required. Use <**Tab**> to move from field to field.

   The Status, Revision Index, Revision Notes, and Name fields are free format. You can enter information in these fields to suit your requirements. The date field has a fixed format and is updated automatically. The information that you enter in the Revision Handling dialog box is used on the cover page of the documentation package when it is printed.

3. When you are finished with this dialog box, select **OK**, and press **Accept/Save**.

This displays the Spool Monitor that shows the printout sequence for the active documents in the package, Figure 3-101. You can change the status of any document at this point if necessary.



```
Spool Monitor (Generation of PDB-Package)

S          OBJ_ID                 DOC_TYPE                FILENAME
-----------------------------------------------------------------------
X  =LIC19MJ00.I_LIST       Spec. Measurement an  EFP_17701_TEXT
X  =M19ML02.I_LIST         Spec. Measurement an  EFP_20040_TEXT
X  =PM1DRY19M&PI1/1.1      P&I Diagram           PFG_19960_PI_DIA
X  =PM1WET14&PI1/1.1       P&I Diagram           PFG_19961_PI_DIA

Change statusflag if necessary (F=Fail,S=Succeeded,X=Generate)
Count: *4                                      <OSC><DBG>     <Replace>
```

Figure 3-101. Spool Monitor

4. Press **Accept/Save** to start the printout.

The spoolfiles are stored in the subdirectory:

- <Project Name>/doc (C_PROJ.DOC) if the printout is started from the DSTRUC object.
- <Project Name>/usr/<User Name> (C_USR) if the printout is started from an object in another structure.

3.4.6 Configure Reports (Application Procedure for SQL*ReportWriter)

The Structure Builder provides access to SQL* ReportWriter which you can use to write and integrate custom reports for database documentation. These reports are included as options in the **Print** and **Print Type** dialog boxes for their respective objects (**Object** menu in Structure Builder) and can be printed on an individual basis via these menu items, or they can be incorporated into and printed with document packages via the Plant Documentation Builder. This section provides an overview for using SQL* ReportWriter to write custom reports for database documentation. For further information regarding SQL*ReprtWriter, refer to:

- SQL*ReportWriter, Building Reports with SQL*ReportWriter, Version 1.1
- ORACLE Keypad Reference
- SQL*ReportWriter, Reference Manual, Version 1.1.

3.4.6.1 Overview SQL*ReportWriter and AdvaBuild Integration

Working Principals of SQL*ReportWriter (Interactive)

In the interactive mode, the SQL*ReportWriter uses the following steps to modify or create an executable Report.

- The following command is used to load the Report from report source file **<report>.rex** into the ORACLE Database to modify the desired report (only for modification of existing reports - not necessary for new reports):

  ```
  > loadrep  <report>  <user>/<pwd> -
  ```

- The following command is used to start the SQL*ReportWriter and work with the loaded report:

  ```
  > sqlrep  <user>/<pwd>
  ```

- Modify ant test the report via generation and execution items
- Exit the SQL*ReportWriter
- Dump the actual state of report definition from the ORACLE Database to the report source file **<report>.rex** via the following command:

  ```
  > dumprep  <report>  <user>/<pwd>  <report>
  ```

- Start the SQL*ReportWriter again and delete the report from the ORACLE Database.

The operating steps in the SQL*ReportWriter environment are shown in Figure 3-102.

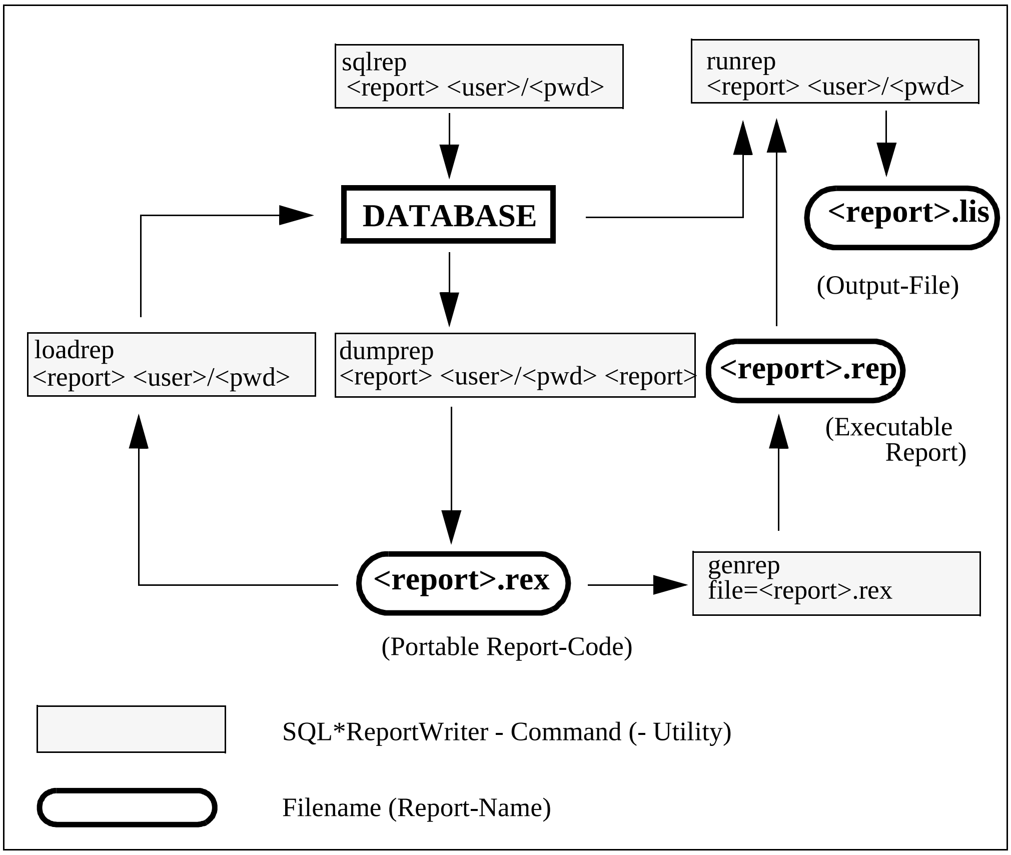


*Figure 3-102. SQL*ReportWriter Overview*

AdvaBuild Integrated SQL*ReportWriter

To design and integrate a report into AdvaBuild (interactive) the following steps are executed:

1. Delete the corresponding Report in the Database (If the report is already stored in the Database, loading of this report is not possible)
2. Load the corresponding Report from **<report>.rex** into the Database
3. Start of SQL*ReportWriter - Design or modify the Report
4. Generate and/or execute the Report (within SQL*ReportWriter) - **<report>.rep**
5. Dump a new File **<report>.rex**
6. Insert or modify a record into the table OBJ_COMMAND to activate this Report.

NOTE

The steps 1, 2 and 4 through 6 are handled automatically by the Plant Documentation Builder.

Overview of Report Print Interface

The Print activation of reports is executed according the following steps:

1. Select administration and formatting values from the AdvaBuild Database
2. Start the desired report with parameters delivered from Structure Builder.
3. The report result is stored in an ASCII file.
4. The ASCII will be converted to PostScript format and added to the corresponding PostScript Footer Template.
5. Footer Parameters will be selected from the AdvaBuild Database and filled in into the footer variables.
6. The final result will be stored into the corresponding directory and printed.

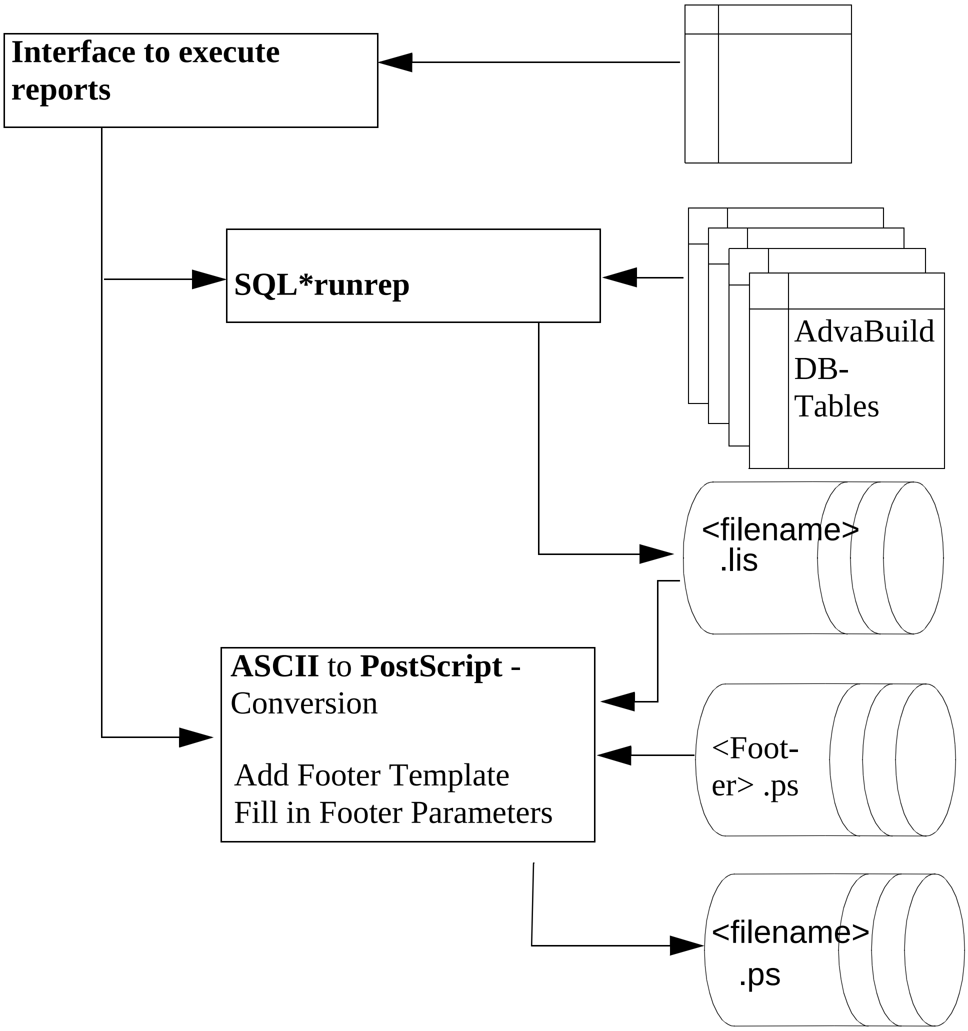


Figure 3-103. Print Sequence of Report-Execution

Basic Elements of Reports

The following is a brief description of some basic elements of a report in SQL*ReportWriter:

- Queries are a means to retrieve data from the database.
- Fields represent column expressions from SELECT statement.
- Groups contain a set of fields to describe sections or subsections in the report.
- Summaries are like fields to display totals.
- Texts contain fields, summaries and parameter references combined with literal strings.
- Report object defines page size, margins and so on.
- Parameters are literal values that will be supplied at runtime for standard and report depending use.

The object hierarchy is shown in Figure 3-104.

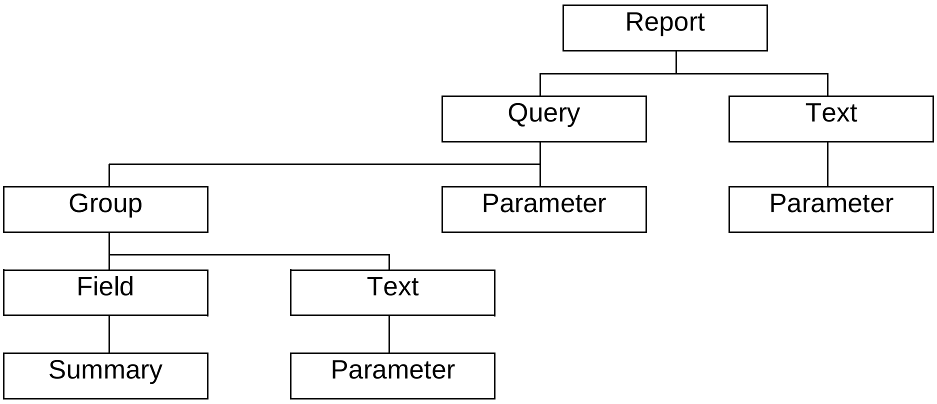


Figure 3-104. Object Hierarchy

3.4.6.2 How to Create and Modify Reports

This procedure associates a specific object type with the report such that when you choose the **Print** or **Print Type** menu item for that object type, the report view will be available as an option in the **Print** (**Print Type**) dialog box.

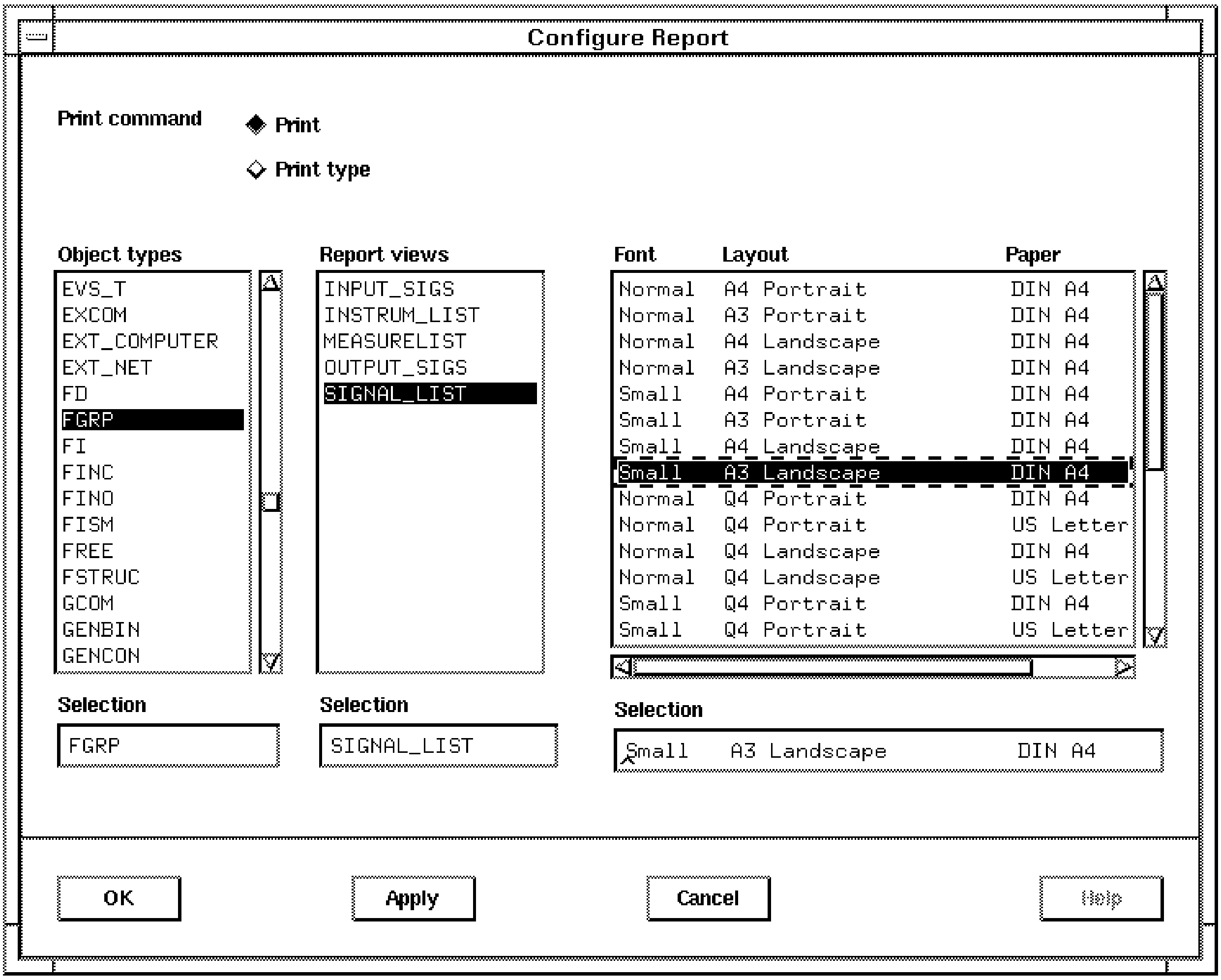
To configure a report:

1. Choose **Configure** from the **Utility** menu in the Structure Builder.

   This displays a pop-up menu with **Configure** options.

2. Choose **ReportWriter** from the **Configure** menu.

This displays the Configure Report Dialog Box, Figure 3-105.



Object types: EVS_T, EXCOM, EXT_COMPUTER, EXT_NET, FD, FGRP, FI, FINC, FINO, FISM, FREE, FSTRUC, GCOM, GENBIN, GENCON

Report views: INPUT_SIGS, INSTRUM_LIST, MEASURELIST, OUTPUT_SIGS, SIGNAL_LIST

Font	Layout	Paper
Normal	A4 Portrait	DIN A4
Normal	A3 Portrait	DIN A4
Normal	A4 Landscape	DIN A4
Normal	A3 Landscape	DIN A4
Small	A4 Portrait	DIN A4
Small	A3 Portrait	DIN A4
Small	A4 Landscape	DIN A4
Small	A3 Landscape	DIN A4
Normal	Q4 Portrait	DIN A4
Normal	Q4 Portrait	US Letter
Normal	Q4 Landscape	DIN A4
Normal	Q4 Landscape	US Letter
Small	Q4 Portrait	DIN A4
Small	Q4 Portrait	US Letter

Selection: FGRP

Selection: SIGNAL_LIST

Selection: Small A3 Landscape DIN A4

Figure 3-105. Configure Report Dialog Box

The Configure Report dialog box is used to get inputs for integration of the desired report into AdvaBuild PRINT / PRINT_TYPE functionality.

3. Select attributes for the corresponding PRINT / PRINT_TYPE command.

These selections create or modify the corresponding object command in the OBJ_COMMAND table for further use by STB menu item **Object Print**. For example:

OBJ_COMMAND			
obj_type	**command**	**com_view**	**pre_descr with**
FGRP	PRINT	SIGNAL_LIST	...<reportname>...<print_format>...

The report defined will have the same name as the object type selected and is bound to this object type.

For any one of the following steps you can either select the attribute from the appropriate list, or enter the required value directly in the corresponding Selection field.

a. Select an Object type (for example, FGRP).

b. Select a report view.

   This is the name used for the report view in the **Print** (or **Print Type**) dialog box.

   A new report view can be created. See Table 3-8.

c. Select a print format.

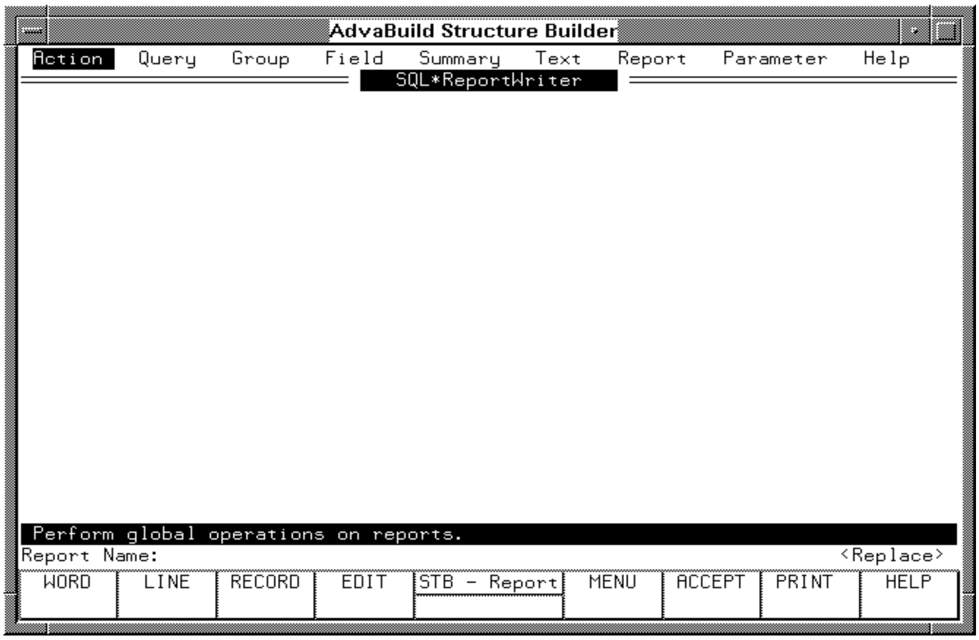
   Only predefined formats can be used. See Table 3-11.

d. Select the object command type (PRINT or PRINT_TYPE).

   Generally, **Print** is for reports that deal with a specific object of the desired type, and **Print Type** is for reports that deal with children of a specific object type.

e. When you are finished with this dialog click on **OK** (or press <**Return**>).

This stores the new report with default definitions in the Oracle database and displays the sqlrep window in global operation mode, Figure 3-106.



Perform global operations on reports.

Report Name: <Replace>

Figure 3-106. sqlrep Window in Global Operation Mode

The most frequently used key functions are described below:

navigate from field to field<**TAB**>, <**ESC**> <**TAB**>

navigate within a fieldRight, Left, Up, Down arrow keys

list of values<**ESC**> <**L**>

accept <**f6**>

cancel <**ESC**> <**f6**>

list of key definitions<**ESC**> <**K**>

replace/insert<**f4**> <.> (. = period key)

For more information see the ORACLE Keypad Reference Manual, page 8-105 to 8-107

NOTES

The Key Assignment on page 8-105 ia partly wrong - Change "f5" to "f4" from Function "mark" to Function "highlight"!

When you use <**ESC**> in combination with another key, press and release <**ESC**>, and then press the other key. DO NOT press and hold <**ESC**>.

Report Naming Conventions

Report names are created according to the following conventions, Table 3-8

Table 3-8. Report Naming Conventions

Object Command	Report Name
PRINT	<Report View>
PRINT_TYPE	<Object Type>_<Report View>

An example is provided in Table 3-9.

Table 3-9. Example, Report Names

Object Type	Object Command	Report View	Report Name
FGRP	PRINT	SIGNAL_LIST	signal_list
FGRP	PRINT_TYPE	MY FGRP	fgrp_my fgrp

Layout

This section describes Corporate Guidelines 9ADA 354 (GATN 61-17-005). In these guidelines data fields are established which are necessary in title blocks for plant documents, as well as their layouts. The layouts are primarily based on the following two documents:

Layout A: DIN 6771, part 5 (modified)

Layout B: Document IEC/DIS 3B (Central Office), in connection with ISO 7200

Paper and Layout format are described in Table 3-10.

Table 3-10. Paper and Layout Format

Abbr.	**Standard**	**Format**	**Width**		**Length**	
			cm	**inch**	**cm**	**inch**
A4	DIN / IEC / ISO	A4	21.000	8.268	29.70	11.693
A3		A3	29.70	11.693	42.00	16.535
-		A2	42.00	16.535	59.40	23.386
U4	ISO	Letter	21.59	8.500	27.94	11.000
-		Legal	21.59	8.500	35.50	14.000
-		Tabloid	27.94	11.000	43.18	17.000
Q4	Merge A + U	Q4	21.000	8.268	27.94	11.000
Q3		Q3	27.94	11.000	42.00	16.535

Orientation can be Landscape (L) or Portrait (P).

The relationship between A4, Q4, and US Letter formats is shown in Figure 3-107.

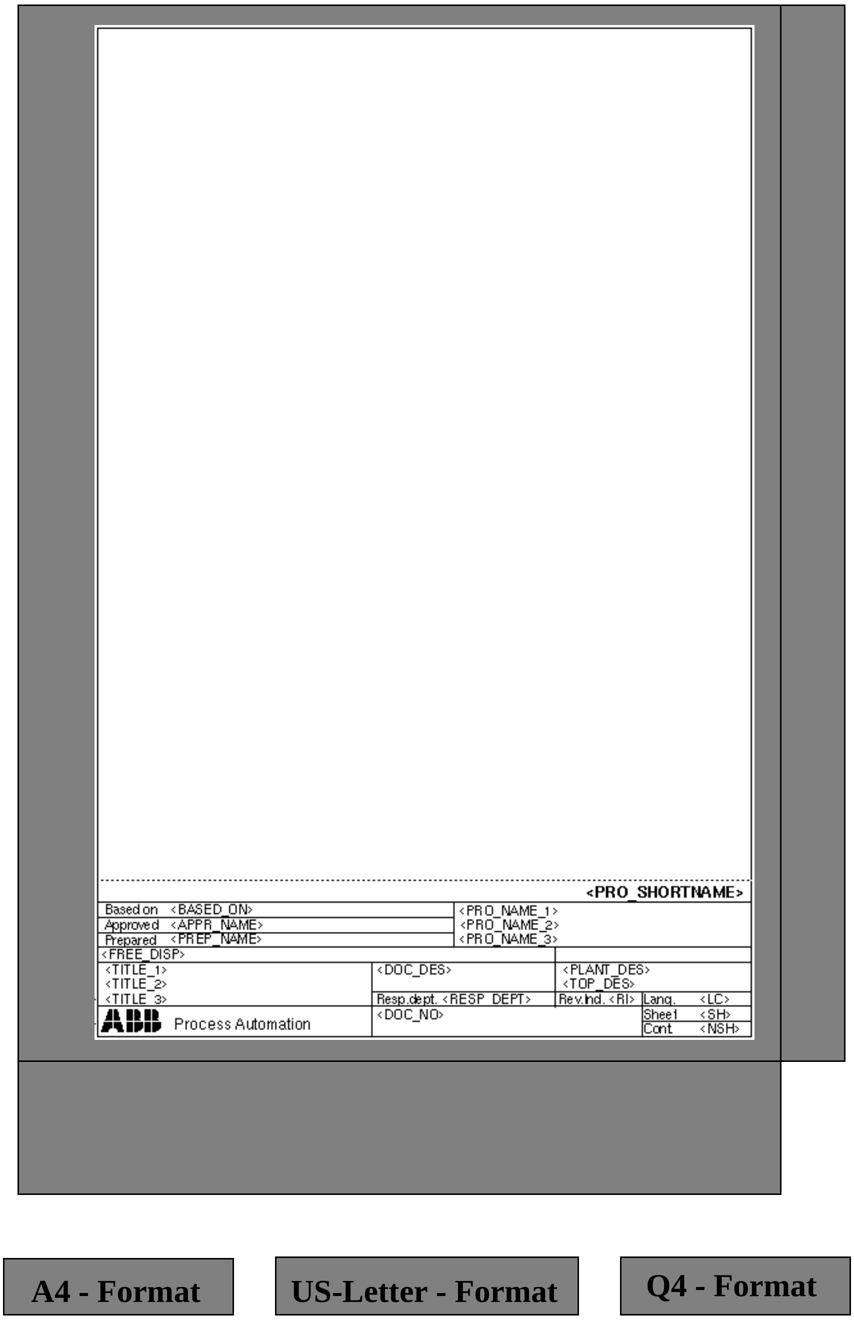


Figure 3-107. Relationship of A4, Q4, and US Letter Formats

Available Print Formats

The print formats described in Table 3-11 can be used in Configure Report dialog box to define the layout of the report to be created or modified.

Table 3-11. Print Formats

Font Descr.	Character Size	Layout	Paper	Mapping
Normal	8 pt	A4 Portrait	DIN A4	a4fa4p
Normal	8 pt	A3 Portrait	DIN A4	a4fa3p
Normal	8 pt	A4 Landscape	DIN A4	a4fa4l
Normal	8 pt	A3 Landscape	DIN A4	a4fa3l
Small	7pt	A4 Portrait	DIN A4	a3fa4p
Small	7pt	A3 Portrait	DIN A4	a3fa3p
Small	7pt	A4 Landscape	DIN A4	a3fa4l
Small	7pt	A3 Landscape	DIN A4	a3fa3l
Normal	8 pt	Q4 Portrait	DIN A4	q4pa4p
Normal	8 pt	Q4 Portrait	US Letter	q4pu4p
Normal	8 pt	Q4 Landscape	DIN A4	q4la4l
Normal	8 pt	Q4 Landscape	US Letter	q4lu4l
Small	7pt	Q4 Portrait	DIN A4	q3pa4p
Small	7pt	Q4 Portrait	US Letter	q3pu4p
Small	7pt	Q4 Landscape	DIN A4	q3la4l
Small	7pt	Q4 Landscape	US Letter	q3lu4l
Small	7pt	Q3 Portrait	DIN A4	q3pa3p
Small	7pt	Q3 Portrait	US Letter	q3pu3p
Small	7pt	Q3 Landscape	DIN A4	q3la3l
Small	7pt	Q3 Landscape	US Letter	q3lu3l
Normal	8 pt	Letter Portrait	US Letter	u4pu4p
Normal	8 pt	Letter Landscape	US Letter	u4lu4l
Small	7pt	Letter Portrait	US Letter	u3pu4p
Small	7pt	Letter Landscape	US Letter	u3lu4l
Small	7pt	Tabloid Portrait	US Letter	u3pu3p
Small	7pt	Tabloid Landscape	US Letter	u3lu3l
Large	12pt	Q4 Portrait	DIN A4	q5fa4p
Large	12pt	Q4 Landscape	DIN A4	q5fa4l

Explanation

Font	**Character size**	**Character per line**		**Number of lines**	
		Portrait	**Landscape**	**Portrait**	**Landscape**
q3xxxx	7 pt	123		97	
q4xxxx	8 pt				
q5xxxx	12 pt	72		56	

3.4.6.3 How to Define a Report

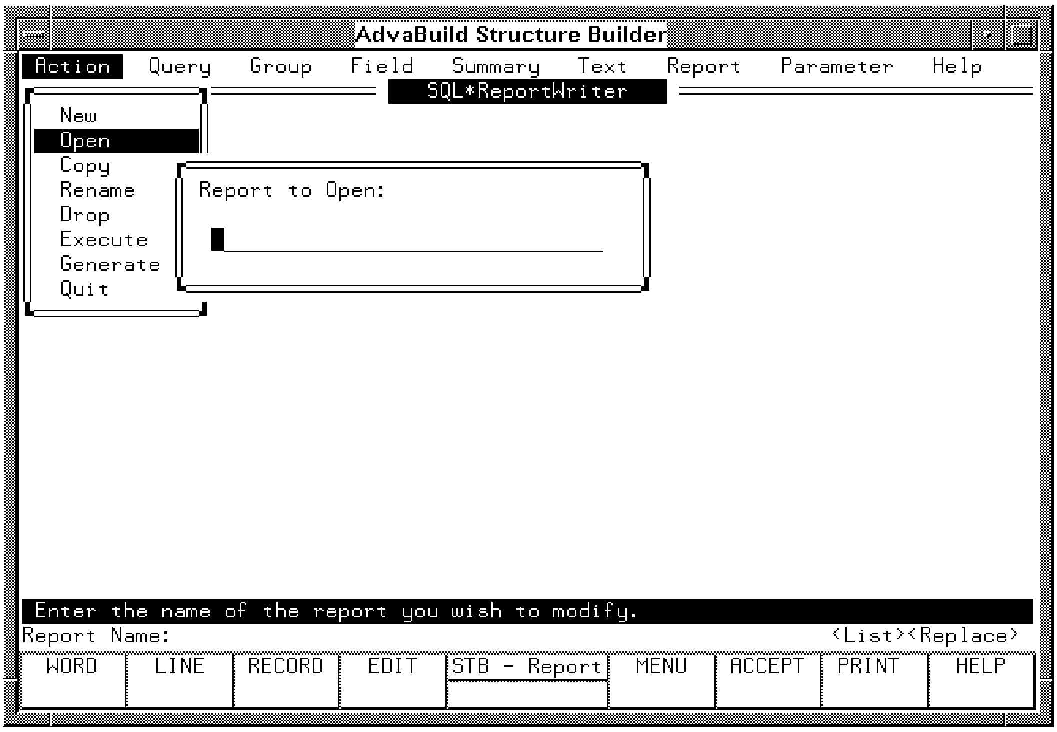
Before you open a report, the report must be configured as described in Section 3.4.6.2, How to Create and Modify Reports. To open a configured report, the desired report with the name according to Section , How to Create and Modify Reports is loaded into the SQL tables of SQL*Report designer:

- The report will be loaded if a report with the computed name exist.
- If the report does not exist a template report will be loaded as computed name report.

The following steps show the sequence of operation to modify / create the report:

1. Choose **Open** from the **Action** menu.

   This displays the Report to open dialog box, Figure 3-108.



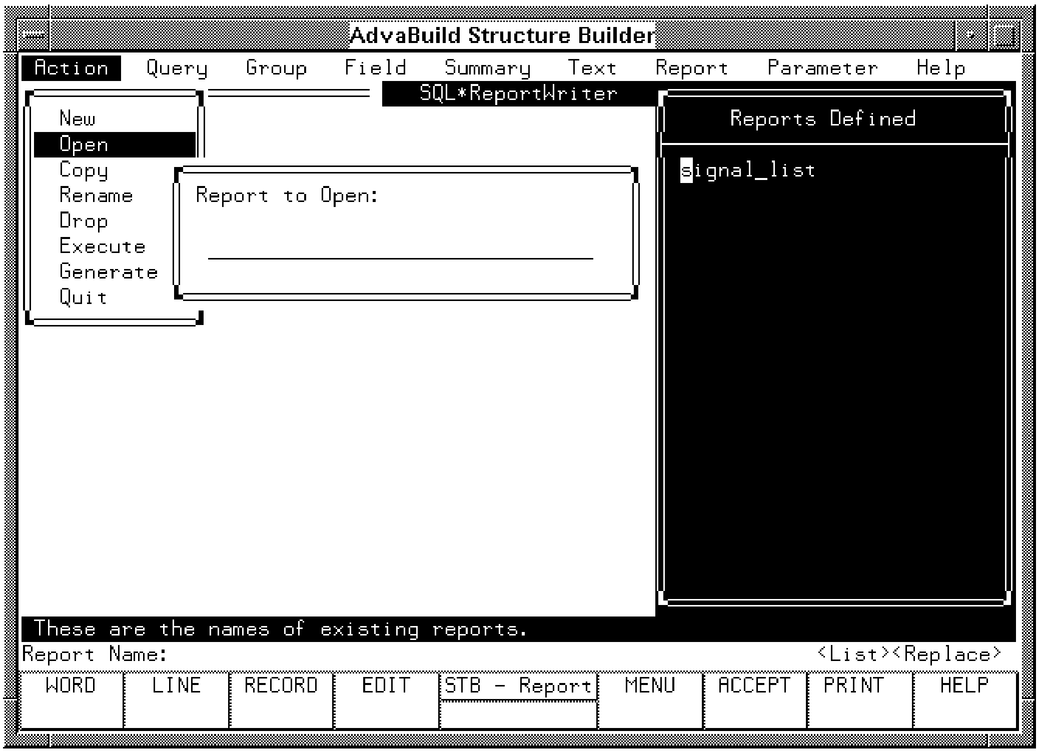
Report to Open:

Enter the name of the report you wish to modify.

Report Name: <List><Replace>

Figure 3-108. Report to Open Dialog

2. Enter the name of the report in the dialog box, or press and release <**ESC**> and then press <**L**> to get a selection box with a list of configured reports, Figure 3-109.



Report to Open:

signal_list

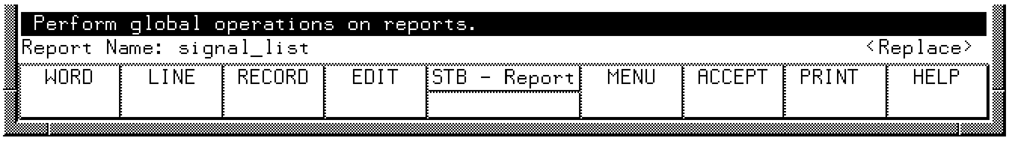
These are the names of existing reports.

Report Name:

Figure 3-109. Open Report Selection Box

3. After you enter the report name, or select a report from the selection box, press <**Return**> to continue.

   The selected report is indicated on the message line, Figure 3-110.



Perform global operations on reports.

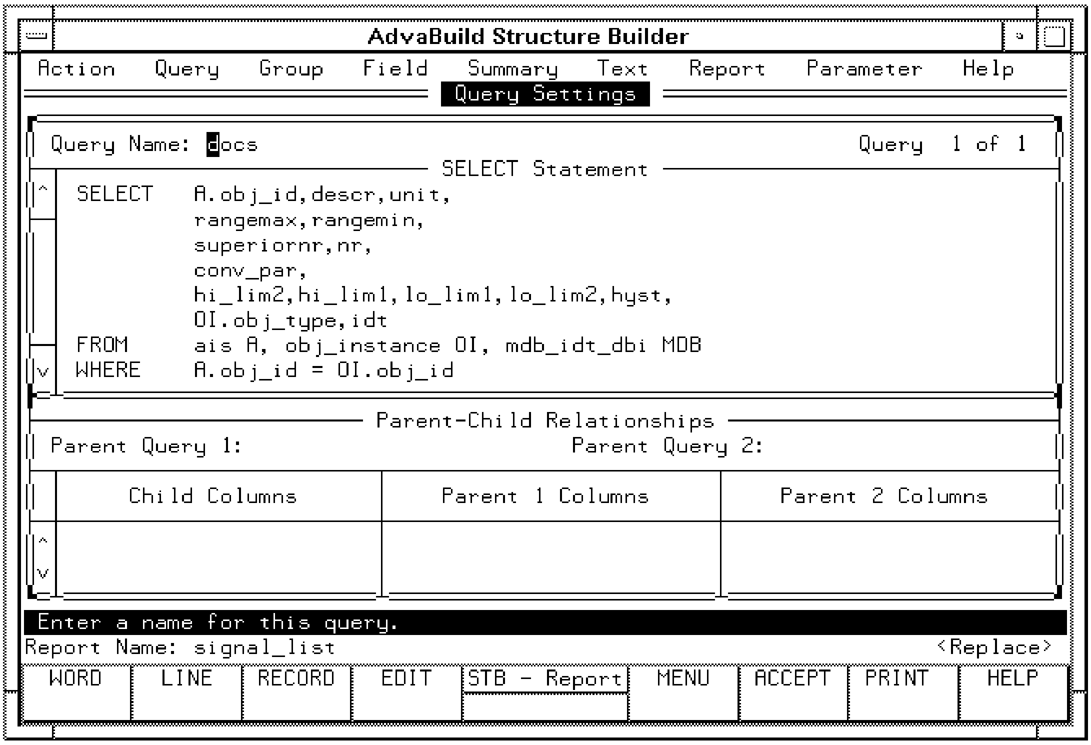
Report Name: signal_list

Figure 3-110. Selected Report Indicated on Message Line

To define a report:

1. Choose **Query** from the sqlrep menu bar.

This displays the Query Settings window with default settings, Figure 3-111.



Query Name: docs   Query 1 of 1

```
SELECT   A.obj_id,descr,unit,
         rangemax,rangemin,
         superiornr,nr,
         conv_par,
         hi_lim2,hi_lim1,lo_lim1,lo_lim2,hyst,
         OI.obj_type,idt
FROM     ais A, obj_instance OI, mdb_idt_dbi MDB
WHERE    A.obj_id = OI.obj_id
```

Parent Query 1:   Parent Query 2:

Enter a name for this query.

Report Name: signal_list

Figure 3-111. Default Query

2. Enter a new query name.

   This step is optional. Typically, you need to enter a unique query name when you are defining more than one query for a given report. Otherwise, you can leave the query name at the default (pseudo).

3. Press <**Tab**> to navigate to the SELECT Statement part of the window.

   This is where you enter a query that defines the contents of the report (what is printed when the report is executed).   The query is based on the SELECT statement which has the following syntax:

   select *attribute1,..., attribute* **from** *table* **where** *condition1* **and &wcond2**

   where:

   attribute1, *attribute* are database attributes

   table is the object type where the selected attributes occur

   condition1 is a condition that must be met to include the selected attributes in the report

   &wcond2 is a mandatory condition

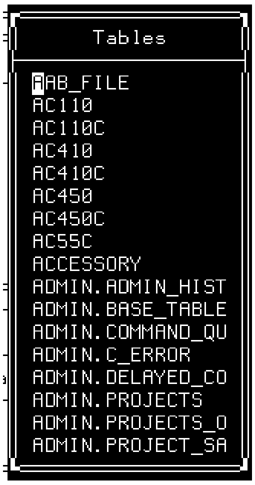
The SELECT statement shown in Figure 3-111 is provided as a default. Edit this statement as required. Some general guidelines are provided below. Consult the SQL*ReportWriter documentation for details.

To define a select statement:

a. Position the cursor in front of **select** where you want to insert an attribute.

b. Press <**ESC**> <**L**>**.**

This displays a list of object types (tables), Figure 3-112.

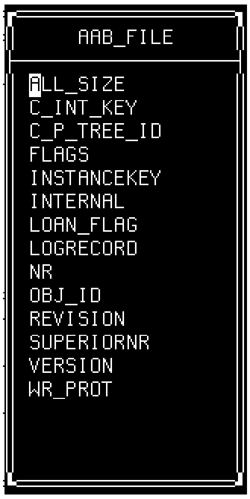


```
Tables
AAB_FILE
AC110
AC110C
AC410
AC410C
AC450
AC450C
AC55C
ACCESSORY
ADMIN.ADMIN_HIST
ADMIN.BASE_TABLE
ADMIN.COMMAND_QU
ADMIN.C_ERROR
ADMIN.DELAYED_CO
ADMIN.PROJECTS
ADMIN.PROJECTS_O
ADMIN.PROJECT_SA
```

Figure 3-112. Object Type Selection Box

c. Position the cursor at the object type that contains the attribute(s) that you want to include in the report and press <**ESC**> <**L**> again.

This displays a list of attributes for the selected object type, Figure 3-113.



AAB_FILE

ALL_SIZE
C_INT_KEY
C_P_TREE_ID
FLAGS
INSTANCEKEY
INTERNAL
LOAN_FLAG
LOGRECORD
NR
OBJ_ID
REVISION
SUPERIORNR
VERSION
WR_PROT

Figure 3-113. Attribute Selection Box

d. Select all attributes you want to include in the report definition by positioning the cursor and pressing **<Return>**. If more attributes are selected than fit on one sheet, you can use SQL*ReportWriter later to adjust the layout.

e. Press <**ESC**> <**f6**> twice in order to quit the selector boxes (first attribute selection box, and then object type selection box).

NOTE

You can delete any default entries from the SELECT statement that are not required for your query.

f. Position the cursor in front of **from** where you want to insert object types.

g. Press <**ESC**> <**L**> to get the list of object types.

h. Select the same object type you selected in step c and press **<Return** >.

i. Press <**ESC**> <**f6**> to exit the selector box.

NOTE

You can delete any default entries from the FROM statement that are not required for your query.

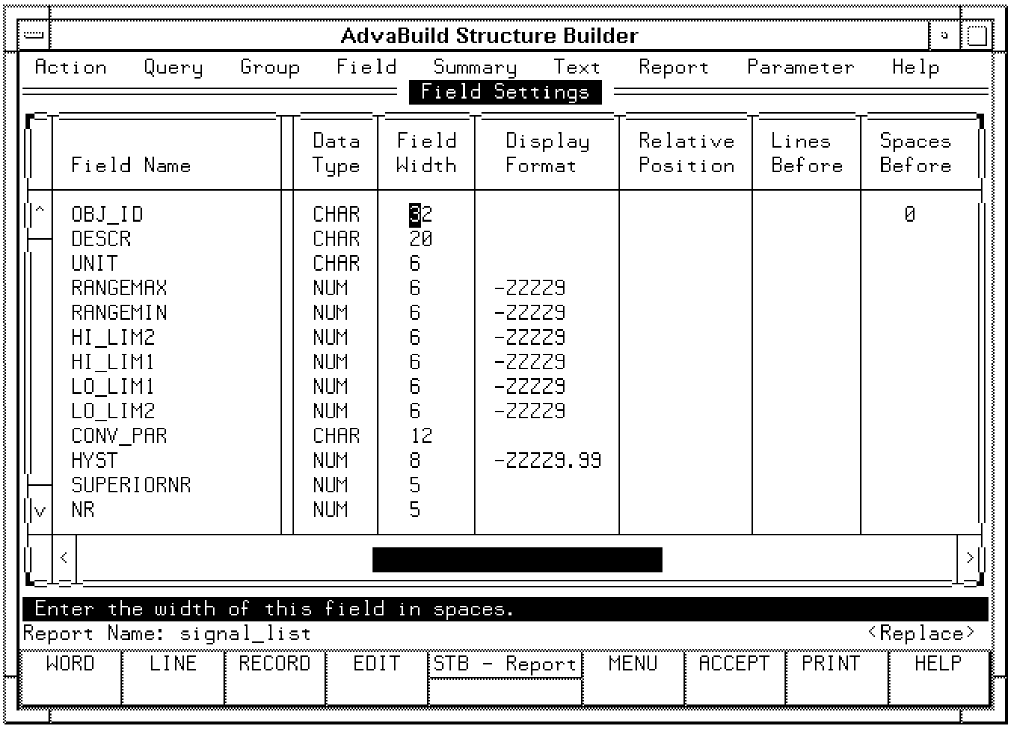
j. Enter any query conditions that you require after **where**.

NOTE

You can delete any default entries from the WHERE statement that are not required for your query. Note that **&wcond2** is a required parameter. DO NOT delete this parameter.

k. Press <**f6**> to store the modifications done so far and to return to the main menu.

4. Modify the display format if necessary, Figure 3-114.



Field Name	Data Type	Field Width	Display Format	Relative Position	Lines Before	Spaces Before
OBJ_ID	CHAR	32				0
DESCR	CHAR	20				
UNIT	CHAR	6				
RANGEMAX	NUM	6	-ZZZZ9			
RANGEMIN	NUM	6	-ZZZZ9			
HI_LIM2	NUM	6	-ZZZZ9			
HI_LIM1	NUM	6	-ZZZZ9			
LO_LIM1	NUM	6	-ZZZZ9			
LO_LIM2	NUM	6	-ZZZZ9			
CONV_PAR	CHAR	12				
HYST	NUM	8	-ZZZZ9.99			
SUPERIORNR	NUM	5				
NR	NUM	5				

Enter the width of this field in spaces.

Report Name: signal_list <Replace>

Figure 3-114. Modify Display Format

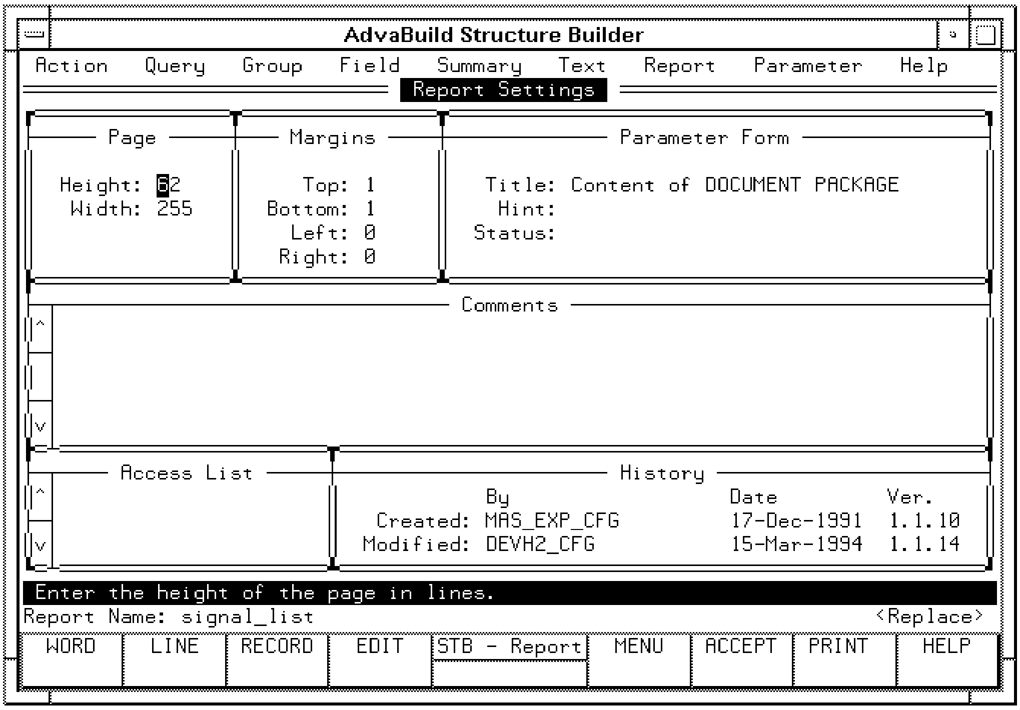
a. Choose **Field** from the sqlrep menu bar (second page).

   This displays the Field Settings window.

b. Modify the Field Width and/or the Display Format (if necessary).

c. Press <**f6**> to store the modifications done so far and to return to the main menu, or press <**ESC**> <**f6**> if nothing was changed.

5. Modify the page size if necessary, Figure 3-115.



```
Height: 62        Top: 1      Title: Content of DOCUMENT PACKAGE
 Width: 255    Bottom: 1       Hint:
                 Left: 0     Status:
                Right: 0

                                   By                  Date          Ver.
                         Created: MAS_EXP_CFG         17-Dec-1991   1.1.10
                        Modified: DEVH2_CFG           15-Mar-1994   1.1.14

Enter the height of the page in lines.
Report Name: signal_list                                         <Replace>
```

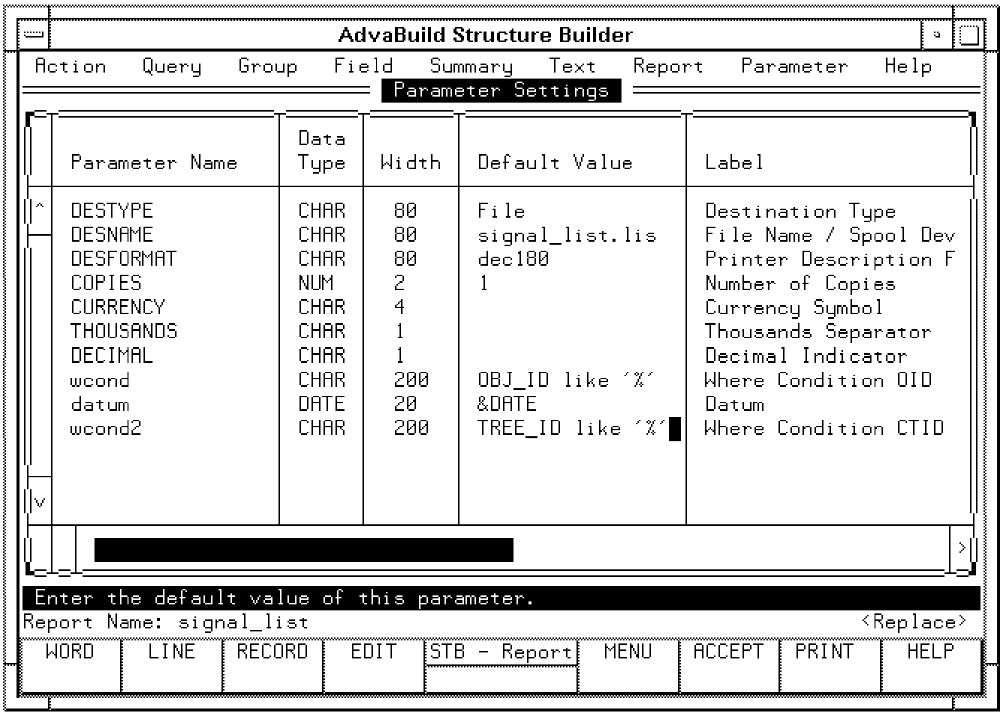
Figure 3-115. Modify Page Size

a. Choose **Report** from the sqlrep menu bar.

b. Check the height and width. The height should be **62** and the width should be **180**.

c. Press <**f6**> to store the modifications done so far and to return to the main menu, or press <**ESC**> <**f6**> if nothing was changed.

6. Modify Report-Parameters if necessary, Figure 3-116.



Parameter Name	Data Type	Width	Default Value	Label
DESTYPE	CHAR	80	File	Destination Type
DESNAME	CHAR	80	signal_list.lis	File Name / Spool Dev
DESFORMAT	CHAR	80	dec180	Printer Description F
COPIES	NUM	2	1	Number of Copies
CURRENCY	CHAR	4		Currency Symbol
THOUSANDS	CHAR	1		Thousands Separator
DECIMAL	CHAR	1		Decimal Indicator
wcond	CHAR	200	OBJ_ID like '%'	Where Condition OID
datum	DATE	20	&DATE	Datum
wcond2	CHAR	200	TREE_ID like '%'	Where Condition CTID

Enter the default value of this parameter.

Report Name: signal_list <Replace>

Figure 3-116. Modify Report Parameters

These Parameters are filled in automatically by Structure Builder and Plant Documentation Builder.

a. Parameter **wcond2** **(Normal use)**

Where - Condition: C_TREE_ID LIKE '%'

(to use for all PRINT_TYPE - Reports -> list all objects of desired Object_Type AINPUT for the selected Subtree)

b. Parameter **wcond**

Where - Condition: OBJ_ID LIKE '%'

(to use for Reports of a single object only).

3.4.6.4 How to Test the Report

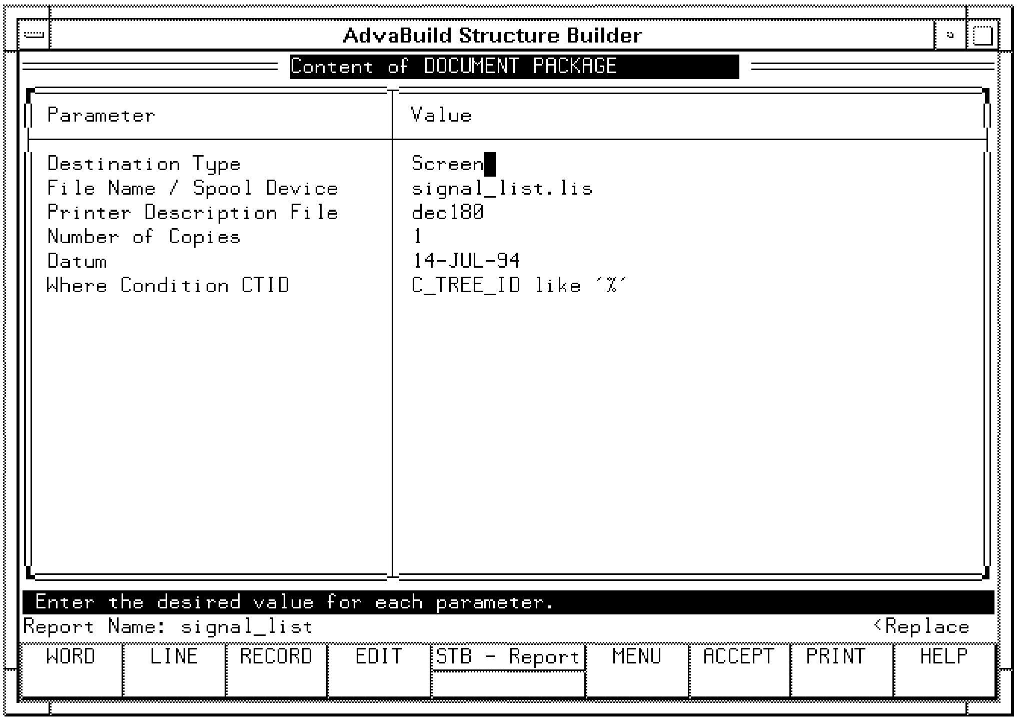
You can test your report definition without leaving the ORACLE Report Writer. In this test mode, footers are not added to the report as they normally are. To test a report:

1. Choose **Execute** from the **Action** menu.

   This displays a dialog box for confirming or canceling execution of the report.

2. To confirm, press <**Return**> in the dialog box.

   This displays the Parameter Values window, Figure 3-117.



Parameter	Value
Destination Type	Screen
File Name / Spool Device	signal_list.lis
Printer Description File	dec180
Number of Copies	1
Datum	14-JUL-94
Where Condition CTID	C_TREE_ID like '%'

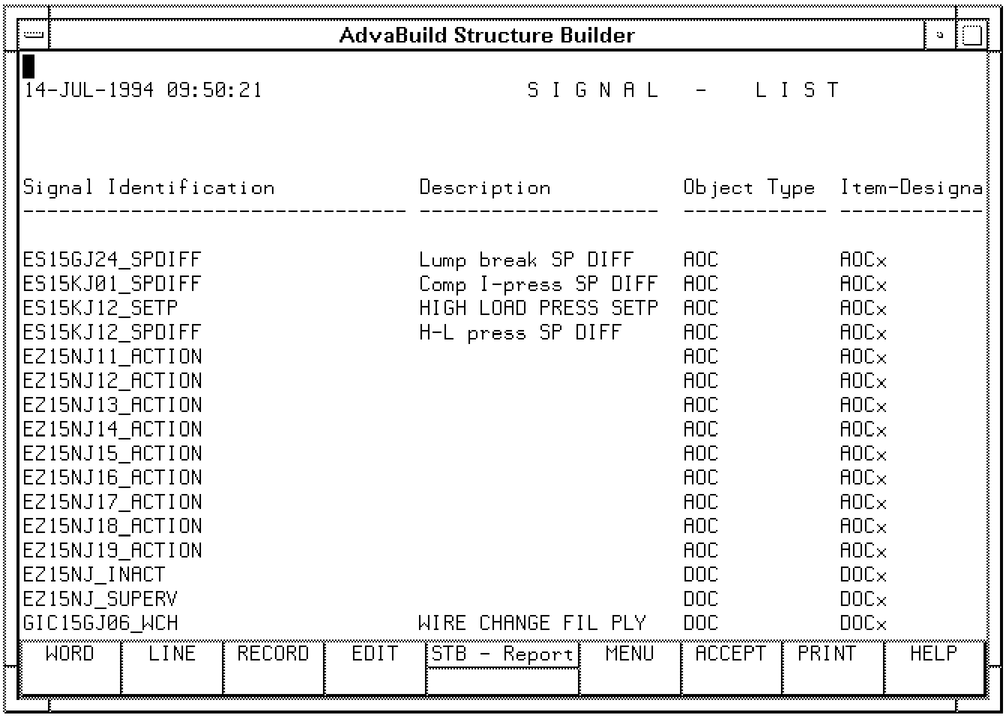
Enter the desired value for each parameter.

Report Name: signal_list <Replace

Figure 3-117. Parameter Values Window

3. Change or enter the parameter values as required and press <**f6**> to execute the report definition.

4. Set the Destination Type to screen to display the result of the report execution on the screen, Figure 3-118.



```
14-JUL-1994 09:50:21                      S I G N A L   -    L I S T



Signal Identification            Description           Object Type  Item-Designa
-------------------------------- --------------------  ------------ ------------

ES15GJ24_SPDIFF                  Lump break SP DIFF    AOC          AOCx
ES15KJ01_SPDIFF                  Comp I-press SP DIFF  AOC          AOCx
ES15KJ12_SETP                    HIGH LOAD PRESS SETP  AOC          AOCx
ES15KJ12_SPDIFF                  H-L press SP DIFF     AOC          AOCx
EZ15NJ11_ACTION                                        AOC          AOCx
EZ15NJ12_ACTION                                        AOC          AOCx
EZ15NJ13_ACTION                                        AOC          AOCx
EZ15NJ14_ACTION                                        AOC          AOCx
EZ15NJ15_ACTION                                        AOC          AOCx
EZ15NJ16_ACTION                                        AOC          AOCx
EZ15NJ17_ACTION                                        AOC          AOCx
EZ15NJ18_ACTION                                        AOC          AOCx
EZ15NJ19_ACTION                                        AOC          AOCx
EZ15NJ_INACT                                           DOC          DOCx
EZ15NJ_SUPERV                                          DOC          DOCx
GIC15GJ06_WCH                    WIRE CHANGE FIL PLY   DOC          DOCx
```

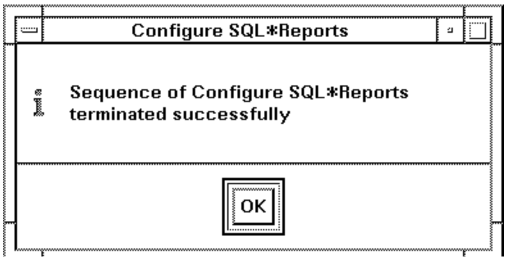
Figure 3-118. Destination Type Screen

The result of the report execution is displayed in the sql output window.

5. Press <**ESC**> <**f6**> to return to the main menu when you are finished.

3.4.6.5 Terminate Session

To exit from the ORACLE Report Writer, choose **Quit** from the **Action** menu. This displays an Information Box to indicate the sequence was successful, Figure 3-119.



Sequence of Configure SQL*Reports
terminated successfully

Figure 3-119. Information Dialog Box

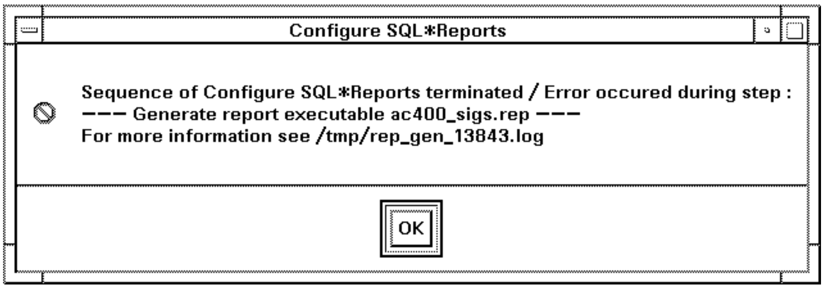
All modifications done so far are saved in a report source file:

“.../<projectname>/rep/<name>.rex”.

In addition, an executable report is generated and stored in a file directory

.../<projectname>/rep/<name>.rep.

If an error occurs you can get more detailed error information in a file listed in the information dialog box, as shown in Figure 3-120.



Sequence of Configure SQL*Reports terminated / Error occured during step :
--- Generate report executable ac400_sigs.rep ---
For more information see /tmp/rep_gen_13843.log

Figure 3-120. Information Dialog Box

3.4.6.6 How to Use the Report in the Plant Documentation Builder

If you want to print the report as part of a documentation package in the Plant Documentation Builder, the report must be inserted in the DOC_RELATION table. Then the report can be handled by the Plant Documentation Builder as any other standard document. Refer to Section 3.4.5, How to Print Plant Documentation for further information.

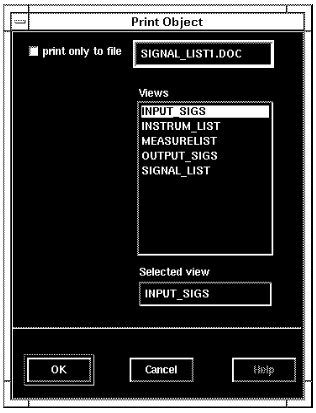
3.4.6.7 Execute a Report

To print out project data you must execute a report which has been defined before. You can execute a report either from the Structure Builder or from the Plant Documentation Builder.

To execute a report from the Structure Builder:

1. Identify the subtree within the Structure Builder that project data should be printed from. This is done by selecting the object heading the subtree.
2. Choose **PrintType** (or **Print**) from the STB's **Object** menu.
3. Select the object type that project data should be printed from.
4. Select the report you want to execute.
5. In the File Print dialog box, enter the filename that the result of the report should be written to and click on **OK** (or the **Return** key), Figure 3-121.

The result of the report is stored in a file named according to the entry in the File print dialog box in the current user directory.



Print Object

■ print only to file: SIGNAL_LIST1.DOC

Views:
- INPUT_SIGS
- INSTRUM_LIST
- MEASURELIST
- OUTPUT_SIGS
- SIGNAL_LIST

Selected view: INPUT_SIGS

Figure 3-121. File Print Dialog Box

The execution of a report within the Plant Documentation Builder is controlled by the Plant Documentation Builder itself. Refer to section 4.7 Operation Instructions/Plant Documentation Builder.

3.4.6.8 Standard Footer

A common footer is provided for project data printed by executing a report. The footer is predefined and need not to be defined by the user. The footer is derived from the Corporate Guidelines of ABB Process Automation [9ADA 354], Figure 3-122.

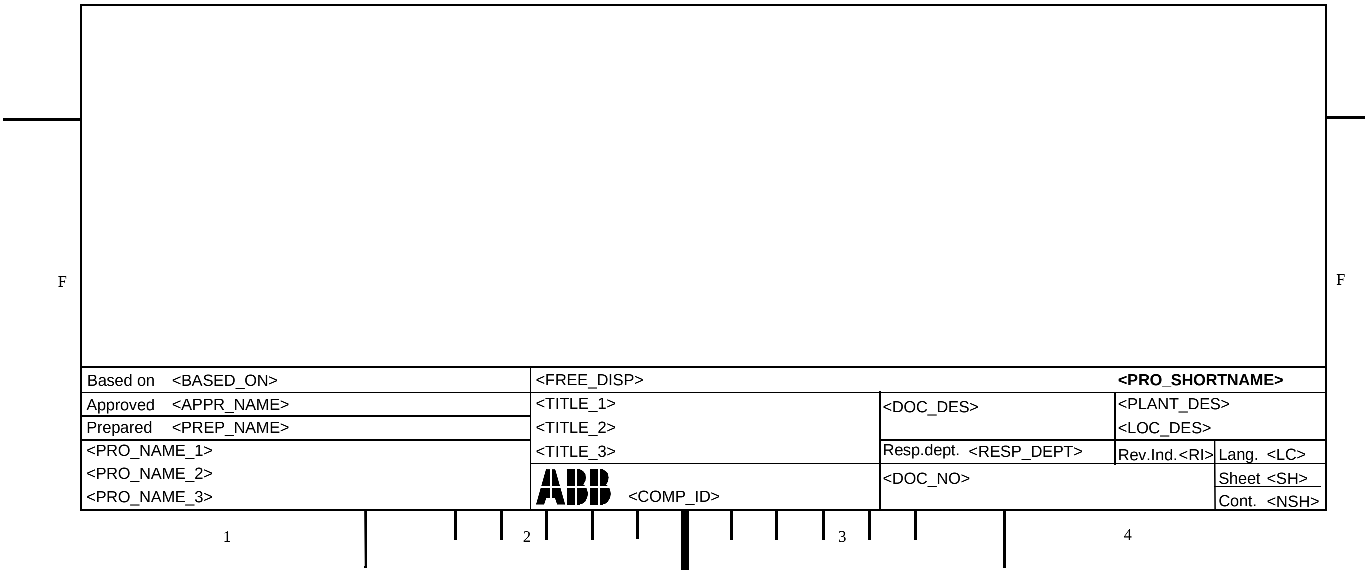


Figure 3-122. Footer Layout (Format A3 Portrait)

The footer fields are described below:

<BASED_ON>	Not yet supported
<COMP_ID>	Company Identification: Name of legal owner, derived from: OWNER attribute of DSTRUC object or OWNER attribute of PROJECT object (Priority DSTRUC - if defined)
<APPR_NAME>(*)	Indicates who approved the document, taken from APPR_NAME attribute of DSTRUC object.
<PREP_NAME>(*)	Indicates who prepared the document, taken from PREP_NAME attribute of DSTRUC object
<PNAM1,2,3>	Project name, taken from PRO_NAME1/2/3 attribute of PROJECT object
<FREE_DISP>(*)	Free disposition, taken from FREE_DISP attribute of DSTRUC object
<PSHNAM>	Project short name, taken from PRO_SHORTNAME attribute of PROJECT object
<TITLE_1,2,3>(*)	Free title of the document, taken from TITLE_1/2/3 attribute of DSTRUC object
<DOC_DES>(*)	Document designation, taken from DOC_DES attribute of DSTRUC object
<RESP_DEPT>(*)	Responsible department, taken from RESP_DEPT attribute of DSTRUC object
<DOC_NO>(*)	Identification of the document, taken from OBJ_ID attribute of DSTRUC object (e.g. 3BDS001099)
<PLANT_DES>(*)	Designation of the plant (functional item designation), taken from function oriented structure depending on the subtree to be printed. The object identity heading the subtree is entered as "PLANT_DES"
< LOC_DES>(*)	Designation of the plant (topographical item designation), taken from: location oriented structure depending on the subtree to be printed. The object identity heading the subtree is entered as "LOC_DES"
<RI>(*)	Revision index entered in a dialog in the Plant Documentation Builder when a document package is generated
<LC>(*)	Language code, taken from LC attribute of DSTRUC object
<SH>	Sheet number, automatically calculated
<NSH>	Number of sheets, automatically calculated

*Fields marked with an asterisk are only applicable for reports that are activated from the Plant Documentation Builder. They are not applicable when the report is activated via the **Print** or **Print Type** commands.

3.4.7 Application Procedures for Solutions

3.4.7.1 How to Use Solutions

A Solution is a structuring concept that provides the means to group objects into units (solutions) that have a certain significance within the engineering process. For example, objects belonging to a certain function or an agglomeration of objects at a certain location may be grouped in a solution.

Objects in a Solution are called **Solution aspect objects**. The Solution itself is represented by a unique object of type SOLUTION. SOLUTION objects reside under a SOLUTIONPOOL object which is a child of the PROJECT object. There can be any number of SOLUTIONPOOL objects under a PROJECT object, and any number of SOLUTION objects under a SOLUTIONPOOL object.

An object becomes a Solution aspect object by inserting it into the **Solution Aspects window** or by linking it to the Solution. The Solution Aspects window provides an overview and direct access to all aspect objects of a Solution.

How to Insert SOLUTION Objects

To insert SOLUTION objects, Figure 3-123:

1. Insert a SOLUTIONPOOL object as a child of the PROJECT object.
2. Insert a SOLUTION object as a child of the SOLUTIONPOOL object.

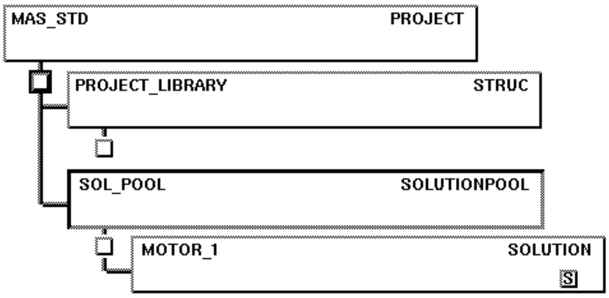
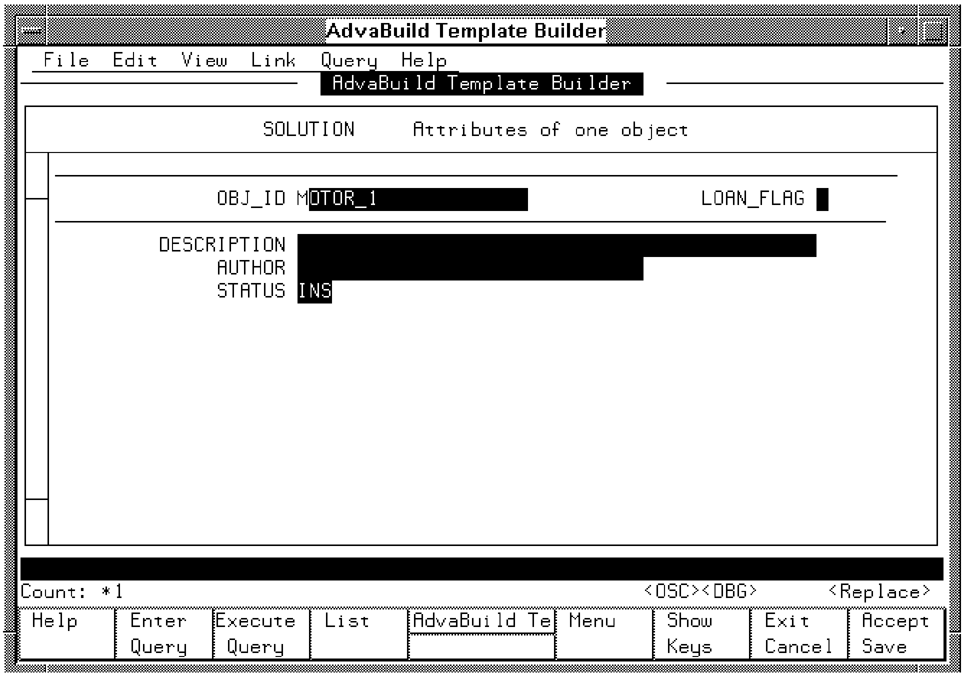


Figure 3-123. SOLUTION Object Beneath a SOLUTIONPOOL Object

How to Open Solution Objects and Define Object Attributes

When you open the SOLUTION object the Open Object dialog box shows two views: PARAMETERS and TEMPLATE. To open the TEMPLATE view for a SOLUTION object, double-click on TEMPLATE. This opens the Template Builder view of the SOLUTION object, Figure 3-124.

You can configure a descriptor (250 characters), and author (30 characters). The STATUS attribute is not configurable and is for internal use only. The PARAMETERS view is not applicable for Master system users.



SOLUTION Attributes of one object

OBJ_ID MOTOR_1
LOAN_FLAG
DESCRIPTION
AUTHOR
STATUS INS

Count: *1 <OSC><DBG> <Replace>

Figure 3-124. Template Builder View of Solution Object

.How to Access the Solution Aspects Window

To access the Solution Aspects window for a specific Solution, click the Solution state button on the SOLUTION object, or the Solution state button on one of the aspect objects for the Solution, Figure 3-125.

As an alternative, you can select the SOLUTION object or any other object belonging to the Solution and then choose **Aspects** from the **View** menu.

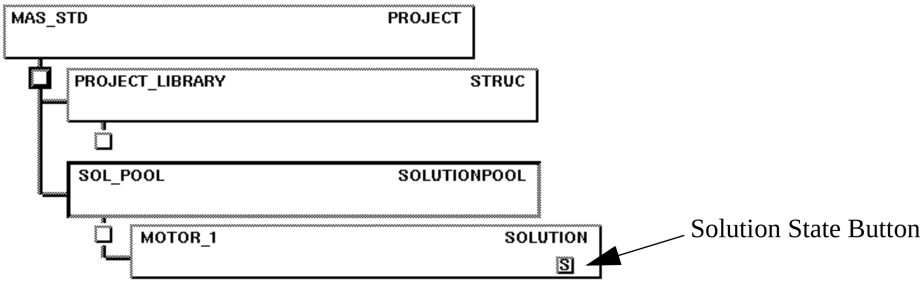
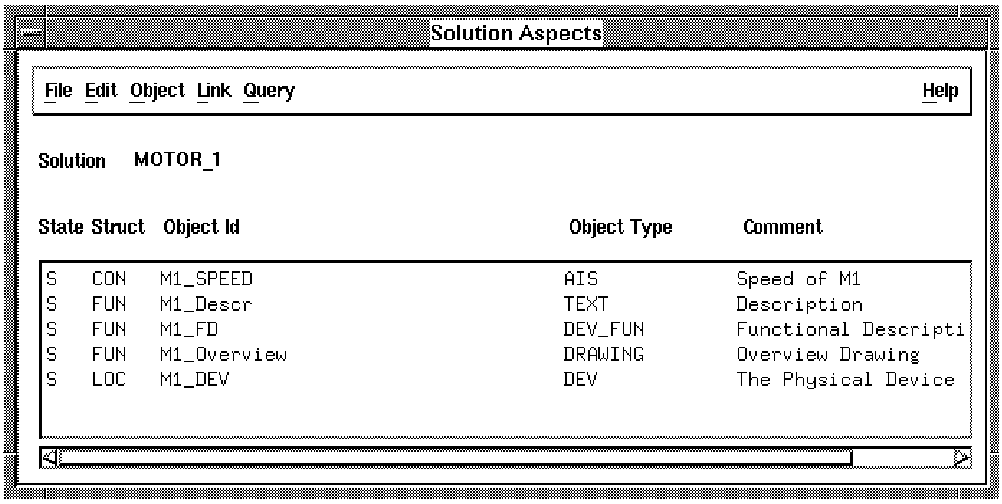


Figure 3-125. Solution State Button

The Solution Aspects window, Figure 3-126, provides an overview and direct access to all aspect objects for a selected Solution. The menus for the Solution Aspects window provide a subset of the functionality provided by the equivalent menus on the Structure Builder menu bar (List Objects windows). For instance you can insert, delete, and rename objects. You can print screen dumps of the Template Builder view, and you can open the Object Information window for an object.



Solution MOTOR_1

```
State Struct  Object Id              Object Type   Comment
S     CON     M1_SPEED               AIS           Speed of M1
S     FUN     M1_Descr               TEXT          Description
S     FUN     M1_FD                  DEV_FUN       Functional Descripti
S     FUN     M1_Overview            DRAWING       Overview Drawing
S     LOC     M1_DEV                 DEV           The Physical Device
```

Figure 3-126. Solution Aspects Window

The Solution field indicates to which Solution the objects in the object list belong to. All aspect objects for the Solution are displayed in a list where each row corresponds to an object. The list has the following columns:

- State

  State indicates the Solution state of the object (as indicated on the Solution state button).

  For Master system users the Solution state always is “S”.

- Struct

  Struct indicates the structure where the object resides:

  - CON: Control-system-oriented structure
  - FUN: Function-oriented structure
  - LOC: Location-oriented structure

- Object ID

  This is the object ID.

- Object Type

  This is the object type.

- Comment

  This is the comment (if any).

How to Add Solution Aspect Objects

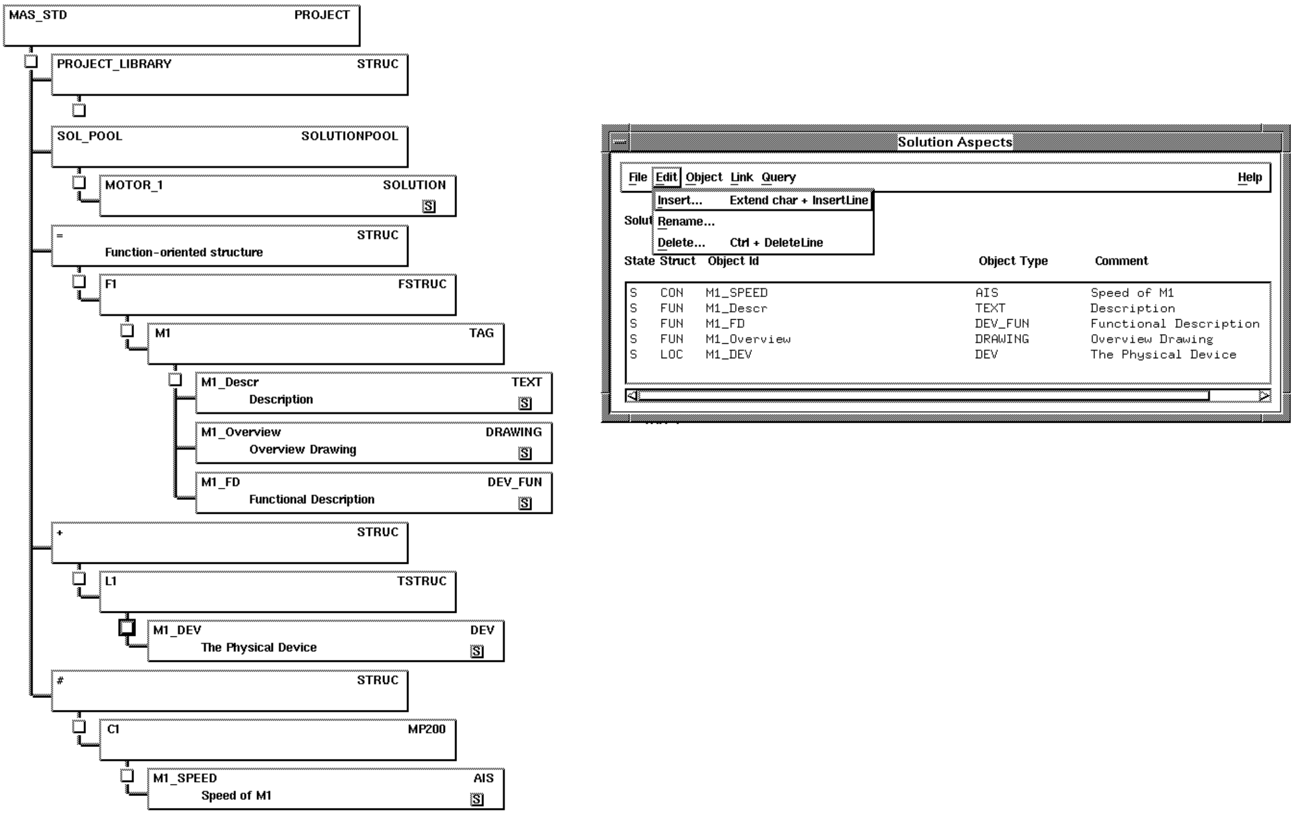
You can add aspect objects to a Solution by either one of the following methods:

- Insert the required object via the Solution Aspect window, Figure 3-127.
- Insert objects via the Structure Builder or Template Builder, and link them (use AdvaBuild Structure Builder - Link Handling) to the SOLUTION object via REFERENCES_SOLUTION links.

The Solution State for the added objects will be set to “S”.

NOTE

An object can only belong to one Solution.



State	Struct	Object Id	Object Type	Comment
S	CON	M1_SPEED	AIS	Speed of M1
S	FUN	M1_Descr	TEXT	Description
S	FUN	M1_FD	DEV_FUN	Functional Description
S	FUN	M1_Overview	DRAWING	Overview Drawing
S	LOC	M1_DEV	DEV	The Physical Device

Figure 3-127. Solution Aspect Objects Spread Out Over the AS 500ES Structures

How to Delete Solution Aspect Objects

You can delete an aspect object for a Solution by either one of the following methods:

- Delete the aspect object via the Solution Aspect window. Multi-selection is supported in Solution Aspects windows.
- Delete the aspect object from the database structure via the Structure Builder.

How to Delete Solutions

When you delete a Solution, you can delete just the Solution and retain the aspect objects, or you can delete the Solution with all of its aspect objects.

To delete a Solution and its aspect objects, first delete all Solution aspect objects via the Solution Aspect window, and then delete the SOLUTION object.

To delete the Solution and retain the aspect objects for that Solution, simply delete the SOLUTION object. The Solution state buttons will be removed from all of the Solution's aspect objects.

3.4.8 How to Integrate and Access the Display Builder

The Display Builder software is a menu driven, interactive utility program used to plan and design custom graphic displays for an Advant station 500 series Operator station of an ABB Master System. The Display Builder software can run in either the operator station or engineering station. A distribution tool is provided to distribute the displays you build to an Operator station for presentation. The Distribution tool copies and moves displays or display libraries to a remote Operator Station, as well as to tape.

This section describes how to integrate the Display Builder software into the AdvaBuild Basic functions so you can access the Display Builder from the Structure Builder. In addition this section describes how to use the distribution tool to distribute displays to an operator station.

For detailed instructions on how to use the Display Builder, refer to the *Display Builder User's Guide*.

3.4.8.1 Handling of Display Objects with Structure Builder

Display objects (object type DISP) are handled via the Structure Builder similar to any other object type. This section provides an overview of the Structure Builder functions that can be applied to display objects, and any considerations that are unique to display objects. Detailed instructions for object handling via the Structure Builder are provided in Section 3.4.2, How to Build the Project Structure via the Structure Builder.

Inserting Display Objects in Engineering Station

Display objects (object type DISP) are inserted into the AdvaBuild database by Structure Builder, Figure 3-128. The parent object must be of type DISPGRP which represents a one-level structure of a display group. The AS object is a child of the STRUC object. DISPGRP objects can only be placed under a AS510OS,AS515OS or AS520OS object.

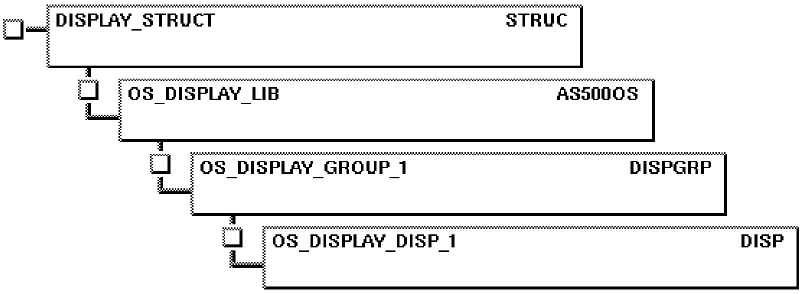


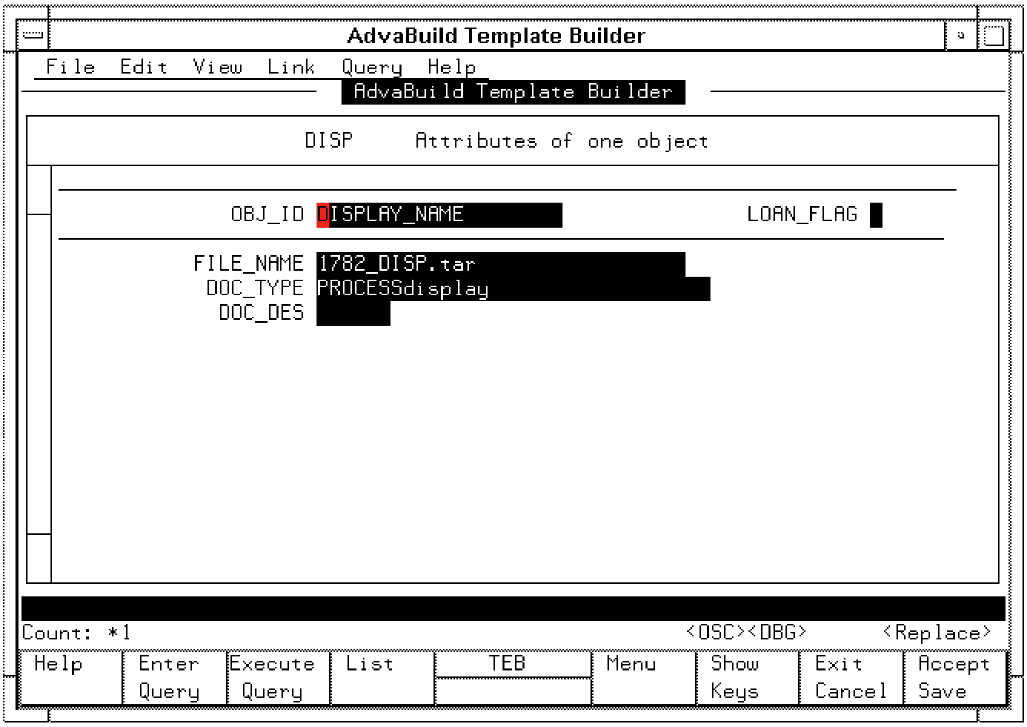
Figure 3-128. Example Object Structure for Display Objects

NOTE

The name of the Display is the object id, and a control of a valid display name is controlled while inserting the new object.

Editing Display Object Attributes

Display objects have attributes that can be defined via the Template Builder, Figure 3-129.


AdvaBuild Template Builder
File Edit View Link Query Help
AdvaBuild Template Builder
DISP Attributes of one object
Count: *1
<OSC><DBG> <Replace>
Help
Enter Query
Execute Query
List
TEB
Menu
Show Keys
Exit Cancel
Accept Save

OBJ_ID DISPLAY_NAME
LOAN_FLAG
FILE_NAME 1782_DISP.tar
DOC_TYPE PROCESSdisplay
DOC_DES

Figure 3-129. DISP Object, template Builder View

- FILENAME

  Physical file name of the display. The value is generated automatically. Its only displayed for information only.

- DOC_TYPE

  Document type of the display. The attribute value combined with the object type is used as key by the DOB for generation of document packages including specific document types. The value is generated automatically. It is displayed for information only.

- DOC_DES

  Designation block for document type corresponding DIN/IEC. For valid values see content of DOC_RELATION table.

Deleting Display Objects

You can delete display objects at the DISP or DISPGRP level. When you delete a DISP object, the corresponding Display object and the files including the display are removed from database and file system. If an object of type DISPGRP is deleted, the DISPGRP object itself is removed and all existing displays within this subtree are also removed.

Copying Display Objects by Structure Builder

Display objects can be copied completely by Structure Builder. This means

- all attribute values are copied.
- the display contents is copied.

Display objects can be copied individually, or as children in a subtree under a DISPGRP object. Objects can be copied under same parent (Predecessor) or to another parent of the right object type. All included objects and their complete contents are copied.

Renaming Display Objects

Display objects and Display group objects can be renamed by the Structure Builder

Moving Display Objects

Display objects can be moved from one display group object to another and display group objects can be moved from one AS5XXOS object to another AS5XXOS by the Structure Builder.

Import Displays

Display objects and Display group objects can be imported from another project.

Printing Displays

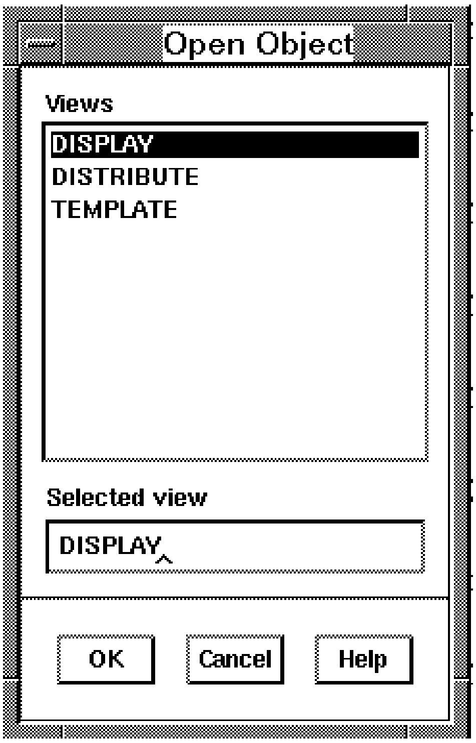
Displays can be printed on color or a black and white postscript printer via the Structure Builder, Display Builder, or Plant Documentation Builder. Refer to the applicable documentation for details.

3.4.8.2 How to Start the Display Builder Via the Structure Builder

To activate the Display Builder from the Structure Builder:

1. Open an existing Display object.

   This displays the Open Object dialog Box for the Display object, Figure 3-130.



Open Object

Views

DISPLAY
DISTRIBUTE
TEMPLATE

Selected view

DISPLAY

Figure 3-130. Open Object Dialog for Display Object

2. Select the Display option.

   What is displayed now depends on whether you are opening this display for the first time or not. If you are opening the display for the first time, a Template dialog box and Display Builder window are displayed, Figure 3-131.

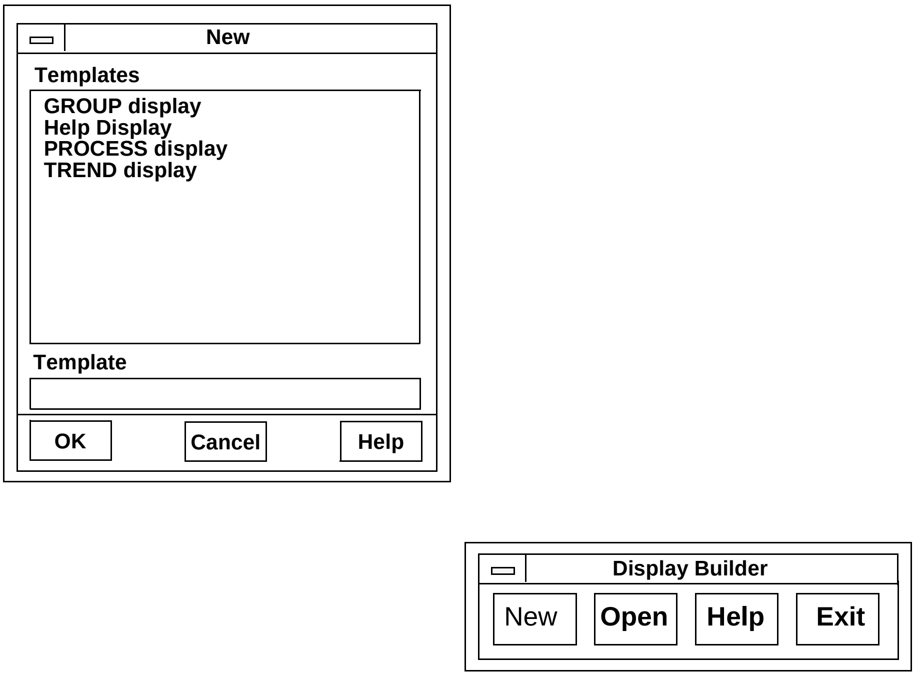


Figure 3-131. Templates Dialog Box and Display Builder Window

To select template type, either double-click, or click once and then click on **OK**.

In a few seconds, a named empty workspace, together with the Colors palette, Graphics box and Display Element Palette are displayed on the screen.

If you have already opened this display before, the selected display with the Color Palette, Graphics Box and Display Element Palette are displayed along with the Display Builder window. In this case, the Template dialog box is not displayed.

In the Display Builder window, the **New** button is dimmed because inserting a new object is performed by Structure Builder. The **Open** button opens an existing display in read-only mode used for cut and paste.

3.4.8.3 Open a Display from Clipboard Directory

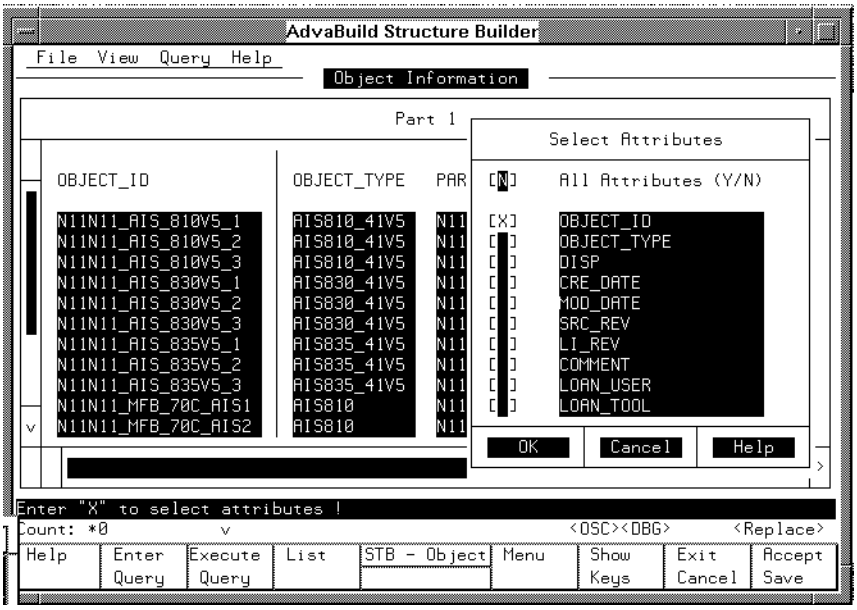
There are two clipboard directories, one for each category standard and adaption. These two directories can be used by the Display Builder for cut and paste purpose. The Distribution tool can also upload displays from an Operator station to those two directories.

1. Start Display Builder from an DISP object.
2. Click on **OPEN** in the Display Builder window.
3. Select **standard** or **adaptions** as category group on Open Dialog box.
4. Select a Display by clicking on the name.
5. Click on **OK.**

3.4.8.4 How to Create an Object List for insertion into DISPLAY BUILDER

Create an Object List of different kind of Object Types

6. Open the Object Information window - Menu: Object, Item: Information - on your wanted project structure level.
7. Select the Enter Query function and enter the appropriated data to get the set of Object Ids you want.
8. Execute Query, the hits will be presented.
9. Select the Menu function to access the Object Information Menu Bar.
10. Choose Print from File Menu.



```
OBJECT_ID              OBJECT_TYPE    PAR
N11N11_AIS_810V5_1     AIS810_41V5    N11
N11N11_AIS_810V5_2     AIS810_41V5    N11
N11N11_AIS_810V5_3     AIS810_41V5    N11
N11N11_AIS_830V5_1     AIS830_41V5    N11
N11N11_AIS_830V5_2     AIS830_41V5    N11
N11N11_AIS_830V5_3     AIS830_41V5    N11
N11N11_AIS_835V5_1     AIS835_41V5    N11
N11N11_AIS_835V5_2     AIS835_41V5    N11
N11N11_AIS_835V5_3     AIS835_41V5    N11
N11N11_MFB_70C_AIS1    AIS810         N11
N11N11_MFB_70C_AIS2    AIS810         N11
```

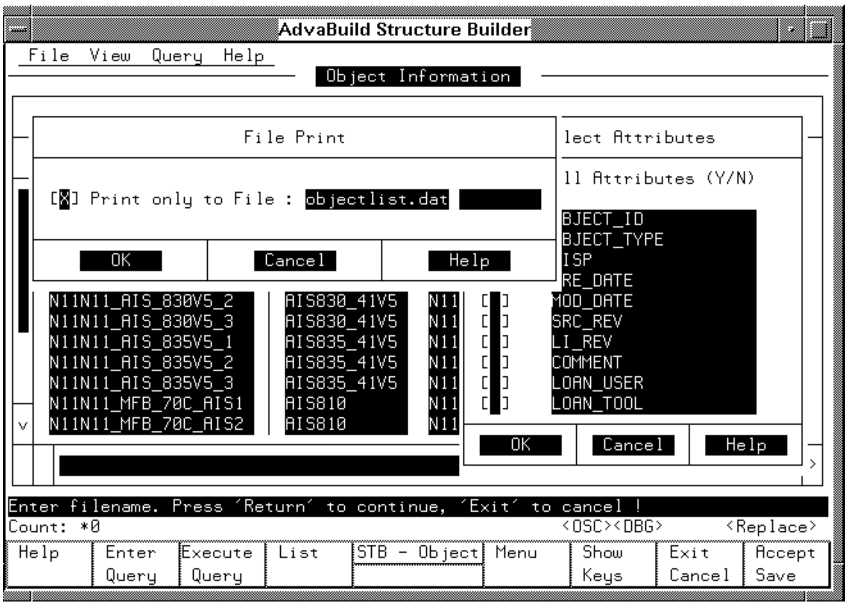
```
[N]   All Attributes (Y/N)

[X]   OBJECT_ID
[ ]   OBJECT_TYPE
[ ]   DISP
[ ]   CRE_DATE
[ ]   MOD_DATE
[ ]   SRC_REV
[ ]   LI_REV
[ ]   COMMENT
[ ]   LOAN_USER
[ ]   LOAN_TOOL
```

```
Enter "X" to select attributes !
Count: *0          v                    <OSC><DBG>     <Replace>
```

Figure 3-132. Creation of an Object List

11. Select on the presented "Select Attributes" Sub Menu:
    a. All Attributes to: "N"
    b. Object ID to: "X"
    c. Press Return.

12. Press Accept/ Save, the Sub Menu "File Print" appears.



```
[X] Print only to File : objectlist.dat
```

```
N11N11_AIS_830V5_2   AIS830_41V5   N11
N11N11_AIS_830V5_3   AIS830_41V5   N11
N11N11_AIS_835V5_1   AIS835_41V5   N11
N11N11_AIS_835V5_2   AIS835_41V5   N11
N11N11_AIS_835V5_3   AIS835_41V5   N11
N11N11_MFB_70C_AIS1  AIS810        N11
N11N11_MFB_70C_AIS2  AIS810        N11
```

```
BJECT_ID
BJECT_TYPE
ISP
RE_DATE
MOD_DATE
SRC_REV
LI_REV
COMMENT
LOAN_USER
LOAN_TOOL
```

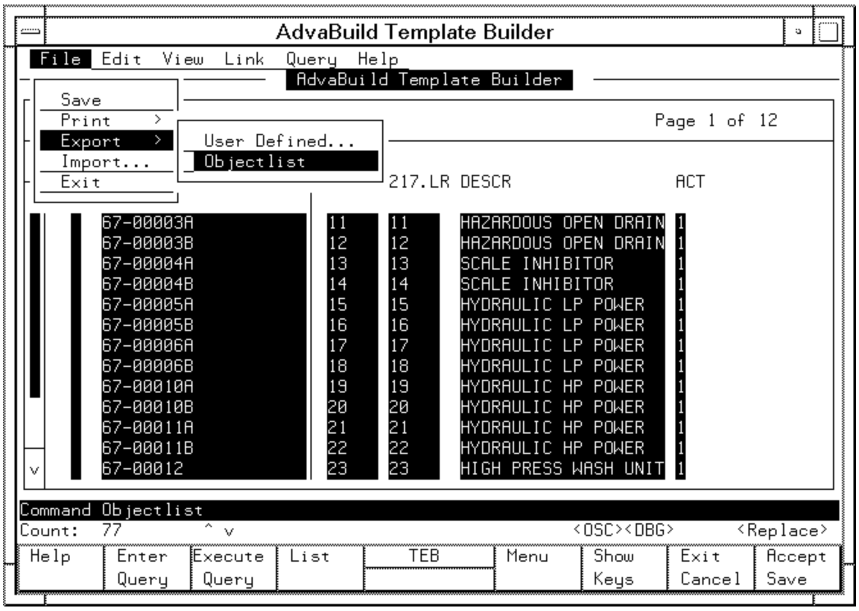
```
Enter filename. Press 'Return' to continue, 'Exit' to cancel !
Count: *0
```

Figure 3-133. Creation of an Object with Print Selection

13. Enter: X "Print only to File" and enter for the file name: **objectlist.dat**
14. Press Return and afterwards press Accept/ Save to execute the print.
15. The procedure to insert the selected Object Ids into the Display Builder see below: "Transfer Object List to Display Builder directory"

Create an Object List of one kind of Object Type

1. Open the **TEB** with the wanted view and Object Type in Multiple Object Form, Figure 3-134.
2. Select the **Menu** function to access the **Users** window menu bar.
3. Choose **Export** from **File** menu and then sub menu **Objectlist**.
4. Press **<Return>**.

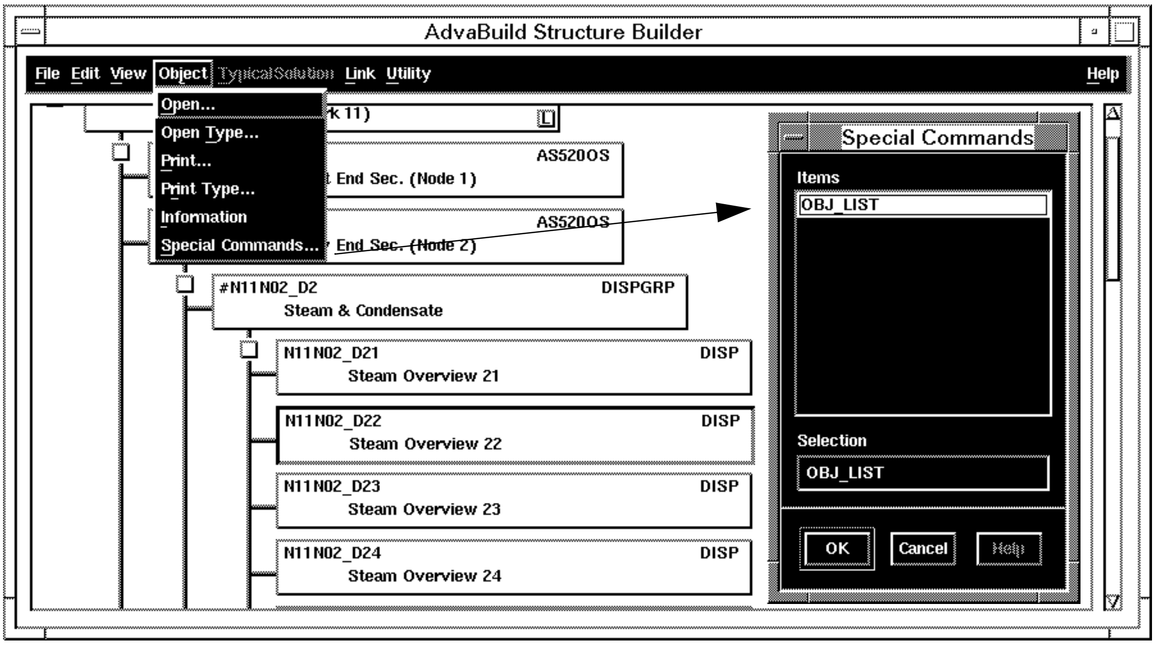


```
AdvaBuild Template Builder
                                                   Page 1 of 12
                                   217.LR DESCR                 ACT
67-00003A          11     11     HAZARDOUS OPEN DRAIN  1
67-00003B          12     12     HAZARDOUS OPEN DRAIN  1
67-00004A          13     13     SCALE INHIBITOR       1
67-00004B          14     14     SCALE INHIBITOR       1
67-00005A          15     15     HYDRAULIC LP POWER    1
67-00005B          16     16     HYDRAULIC LP POWER    1
67-00006A          17     17     HYDRAULIC LP POWER    1
67-00006B          18     18     HYDRAULIC LP POWER    1
67-00010A          19     19     HYDRAULIC HP POWER    1
67-00010B          20     20     HYDRAULIC HP POWER    1
67-00011A          21     21     HYDRAULIC HP POWER    1
67-00011B          22     22     HYDRAULIC HP POWER    1
67-00012           23     23     HIGH PRESS WASH UNIT  1
Command Objectlist
Count:  77        ^ v                          <OSC><DBG>     <Replace>
```

Figure 3-134. Object List with one Kind of Object Type

Transfer Object List to Display Builder directory

1. Select the **DISP** Object with the Display, in which you want use the created Object List later.
2. Choose **Special Commands...** from **Object** menu in STB.
3. Select OBJ_LIST and press **<OK>** button, Figure 3-135.

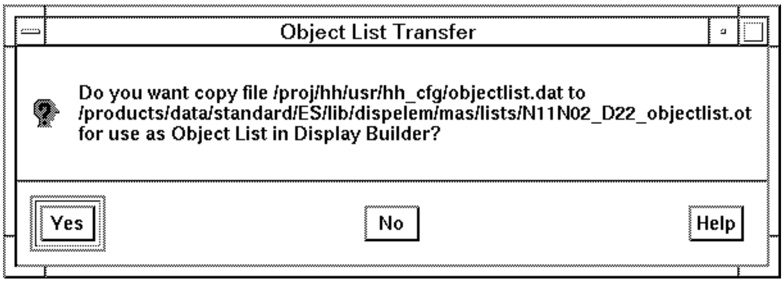


```
k 11)
AS520OS
t End Sec. (Node 1)
AS520OS
/ End Sec. (Node 2)
#N11N02_D2                    DISPGRP
        Steam & Condensate
N11N02_D21                    DISP
        Steam Overview 21
N11N02_D22                    DISP
        Steam Overview 22
N11N02_D23                    DISP
        Steam Overview 23
N11N02_D24                    DISP
        Steam Overview 24
```

Selection: OBJ_LIST

Figure 3-135. Transfer of Object List

4. Press **<Yes>** button for Object List Transfer in the appearing window, Figure 3-134.



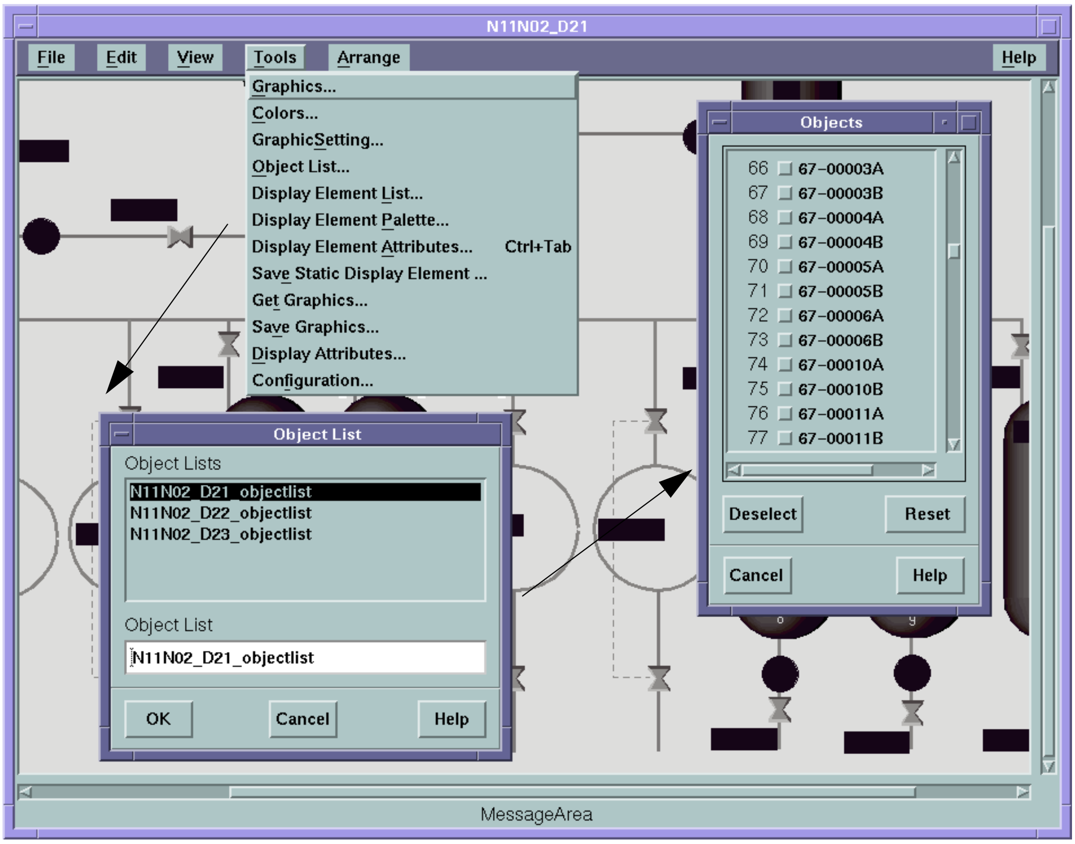
Do you want copy file /proj/hh/usr/hh_cfg/objectlist.dat to /products/data/standard/ES/lib/dispelem/mas/lists/N11N02_D22_objectlist.ot for use as Object List in Display Builder?

Figure 3-136. Object List Transfer Box

5. If a Object List for this Display was already transferred, you will be ask for adding or overwriting of the Object List. Press **<Yes>** or **<No>**.

Use of Object List in Display Builder

1. Select the wanted **DISP** object and start Display Builder.
2. Choose **Object List...** from **Tools** menu in Display Builder, Figure 3-137.
   Now the **Object List** window will appear with all already transferred Object Lists. The name of Object List will be created by transfer with Special Command with the selected OBJ_ID plus extension '_objectlist' (<obj_id>_objectlist).
3. Select the wanted **Object List** and press **<OK>** button.
   Now the **Objects** window with the list of objects will appear for use in Display Builder.



Object Lists
N11N02_D21_objectlist
N11N02_D22_objectlist
N11N02_D23_objectlist

Object List: N11N02_D21_objectlist

66 ☐ 67-00003A
67 ☐ 67-00003B
68 ☐ 67-00004A
69 ☐ 67-00004B
70 ☐ 67-00005A
71 ☐ 67-00005B
72 ☐ 67-00006A
73 ☐ 67-00006B
74 ☐ 67-00010A
75 ☐ 67-00010B
76 ☐ 67-00011A
77 ☐ 67-00011B

Figure 3-137. Use of Object List in Display Builder

3.4.8.5 How to Use the Distributor

To activate the distributor tool, open a DISP or DISPGRP object as described earlier, and select the DISTRIBUTE option in the Open Object dialog box. If you choose to distribute an object of type DISPGRP all displays which are children of this group will be ready for distribution.

When you select the DISTRIBUTE option, the Distributor window is opened, Figure 3-138.

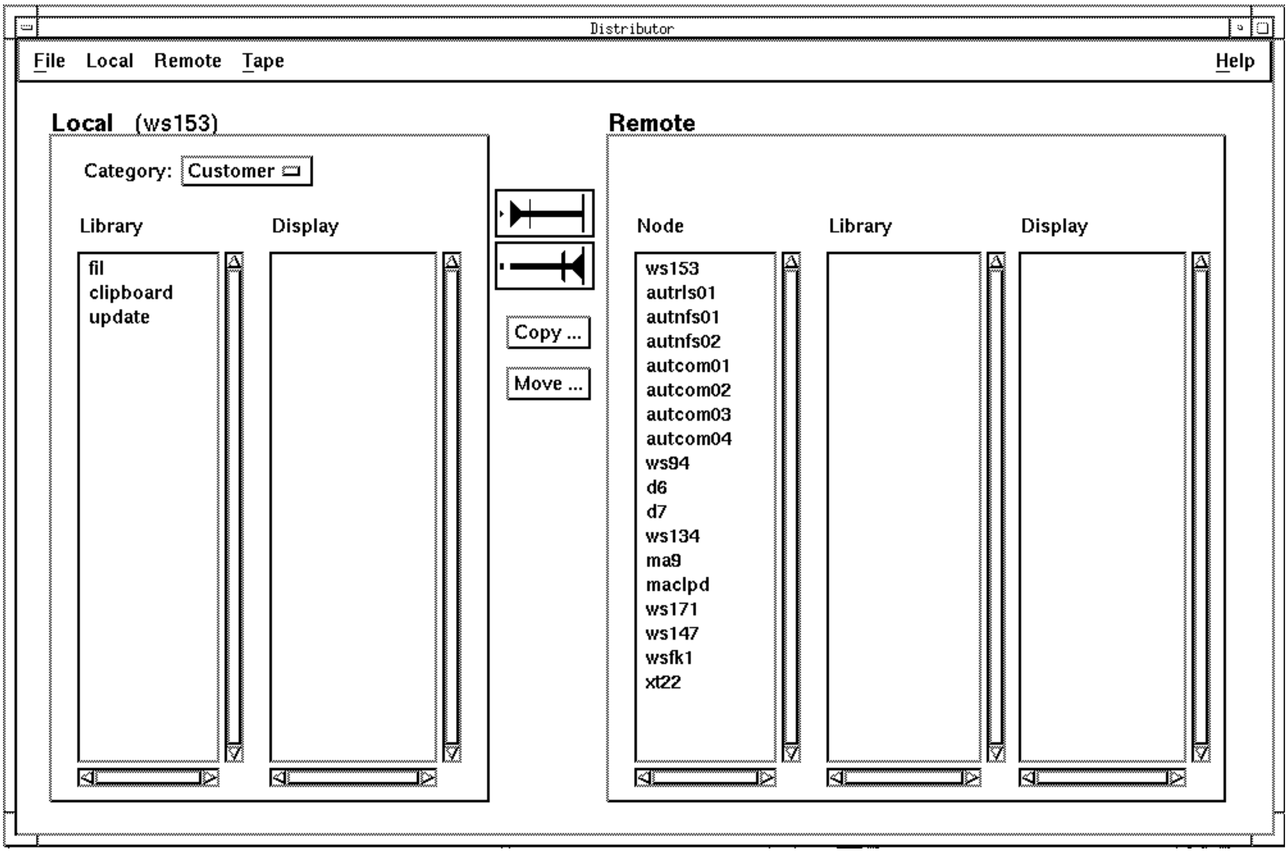


Figure 3-138. Distributor Window

This is the working interface for display distribution. It consists of two major parts, the Local node to the left and the remote node to the right. A category button is situated to the top left on each side and is used to change between the different user categories defined. Besides this the interface contains a menu bar, buttons for the copy and move functions, and arrows which represent the direction of the copy or move operation.

Download a Display or a Group of Displays to an Operator Station

To distribute the chosen display or display group from Structure Builder:

1. Start Distribution tool from a DISP or DISPGRP object.
2. Click at the directory **fil** in the Library column on Local side.
3. Select the display to distribute by clicking on a display in the display column on local side (this is optional if you do not perform this action the library will be highlighted and all displays shown in the display column will be distributed).
4. Select Remote node.
5. Select another category group or library on the destination side (Optional).
6. Select remote library
7. Select **right** arrow.
8. Click on **copy**
9. To exit, choose **Exit** from the **File** menu in the Distributor window.

Upload a Display or a Group of Displays from an Operator Station

You can upload a display or a group of displays from an Operator station and insert them to the Structure Builder tree. All display objects that are placed in the update directory on local side will be inserted in the Structure Builder tree when exiting the Distribution tool. The new inserted objects will be inserted in the same level or under the object where the Distribution tool was started depending on whether this object was of type DISP or DISPGRP. If the object already exists a question dialog box pops ups with a question if you want to update the object with the contents of the new display.

1. Start the Distribution tool at a DISP or DISPGRP object.
2. Click at the directory **update** in the Library column on Local side.
3. Select Remote node.
4. Select another category group on the destination side (Optional).
5. Select remote library
6. Select the display to upload by clicking on a display in the display column on remote side (this is optional if you do not perform this action the library will be highlighted and all displays shown in the display column will be uploaded).
7. Select **left** arrow.
8. Click on **copy**
9. To exit, choose **Exit** from the **File** menu in the Distributor window.

Upload a Displays from an Operator Station to Clipboard Directory

You can upload a display or a group of displays from an Operator station and place them in the clipboard directory.

1. Start Distribution tool from a DISP or DISPGRP object.
2. Select **customer, standard** or **adaptions** as category group on the local side.
3. Click at the directory **clipboard** in the Library column on Local side.
4. Select Remote node.
5. Select another category group on the destination side (Optional).
6. Select remote library
7. Select the display to upload by clicking on a display in the display column on remote side (this is optional if you do not perform this action the library will be highlighted and all displays shown in the display column will be uploaded)
8. Select **left** arrow.
9. Click on **copy**
10. To exit, choose **Exit** from the **File** menu in the Distributor window.

3.4.8.6 Exit Display Builder

To exit Display Builder after building a display:

1. First choose **Save** from the **File** menu in the Display Builder menu bar, and then choose **Close** from the **File** menu, Figure 3-139.

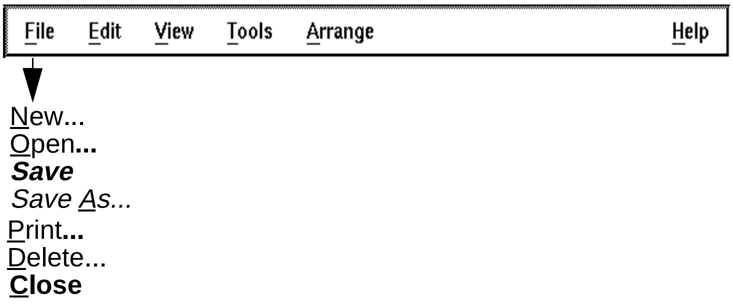


Figure 3-139. Display Builder File Menu

2. Click on the **Exit** button in the Display Builder window,. This exits this instance of the Display Builder and all the windows belonging to this session.

3.4.9 Export/Import Utility

3.4.9.1 Introduction

The Export/Import Utility (EIU) provides the means to:

- import data from an ASCII file into the AdvaBuild database using the ORACLE SQL*Loader.
- export data from the AdvaBuild database into an ASCII file.

This way you can transfer data between Advant Stations when a serial interface is not available.

Audience

The Export/Import Utility uses the ORACLE SQL*Loader to import data. Therefore, you should refer to the ORACLE7 Server Utilities User's Guide for further information. Prior to using this utility, you should be familiar with:

- the operating system of the computer including its file system and file handling utilities
- the SQL language
- the AdvaBuild database and file system.

NOTE

EIU users must be assigned to the CONFIG user access level.

Related Documents

Along with this user's guide, you may want to refer to the following documents:

- ORACLE7 SQL Language Reference Manual
- ORACLE7 Server Utilities User's Guide for the SQL*Loader
- Object Type Definition Manual for the AS500ES database system.

Conventions

Alternative Items: Alternative choices are separated by vertical bars. The set of choices is enclosed by curly brackets if one of the items is required, or square brackets if the item is optional.

Required Items: Required items are enclosed in curly brackets. You must choose one of the alternatives.

{BADFILE | BADDN} filename

Optional Items: Optional items are enclosed in square brackets.

INTEGER [(length)]

Repetitive Items: An ellipsis represents an arbitrary number of similar items.

[columnspec, columnspec...]

3.4.9.2 Basic Terms

Select attribute: This term describes the attribute selected from a source table (see below), i.e. the source attribute where to get necessary input data from.

Source table: If data are exported from the database, this term describes the object-type or link-type tables, from which the EIU exports these data.

Target attribute: This describes the attribute where to write certain data in the target table (see below), i.e. the attribute where to put data received from input data or from the data base.

Target table: The term "target tables" represents the object-type or link-type tables of the AS500ES system, into which data are imported. The support tables of the system cannot be changed with the EIU.

Temporary table: When data are imported into the AS500ES system a temporary table is created within the database first. This table is handled by the EIU and the user only has to specify its table name, which may not be the name of an existing table within the AS500ES database.

3.4.9.3 Overview

Importing data into the database

An overview of the data import process is shown in Figure 3-140.

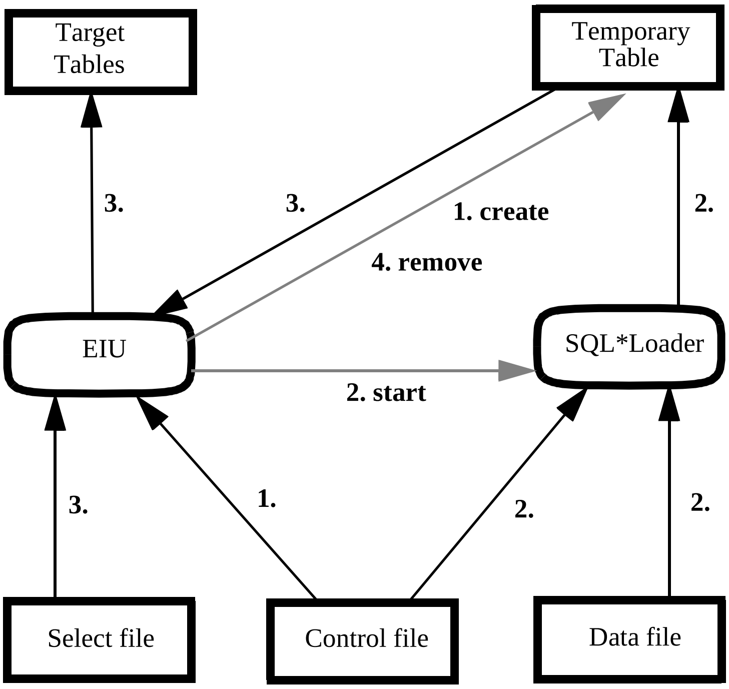


Figure 3-140. Import of Data

Steps during import:

1. The temporary table is created by EIU based on the control file.
2. Data are loaded into the temporary table via SQL*Loader.
3. Data are selected from the temporary table and loaded into the target tables of the database. This selection may be controlled by a select file
4. The temporary table is removed.

If a file is to be stored into the database, first an object is created as described above, then the file is read from the input directory and stored in the database.

Exporting data from the database

An overview of the data export process is shown in Figure 3-141.

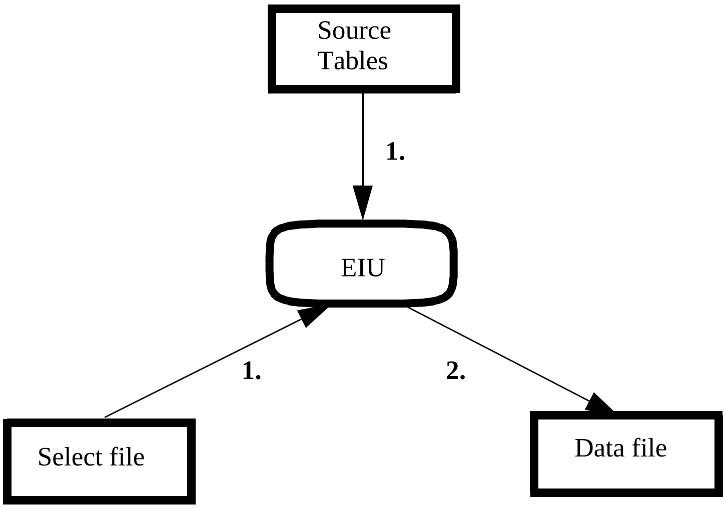


Figure 3-141. Export of Data

Steps during export:

1. Select data from the source tables of the database using the selectfile.
2. Write data into the data file.

If a file is to be extracted from the database, first the object data are written into the data file, then the file is written into the file system.The corresponding object in the database, from which to export the file, is selected with a select file, like in the import of data (Figure 3-140).

Starting EIU

The Export/Import Utility may be started in two different ways:

- By choosing either **Import** or **Export** from the **File** menu in the Template Builder
- From the command line within a terminal window.

Files and Directories

All files handled by the EIU must be placed in the export/import directory within the AdvaBuild directory structure of the corresponding project, Figure 3-142. Within the AdvaBuild environment, this directory is also represented by an environment variable **C_PROJ_EIU.**

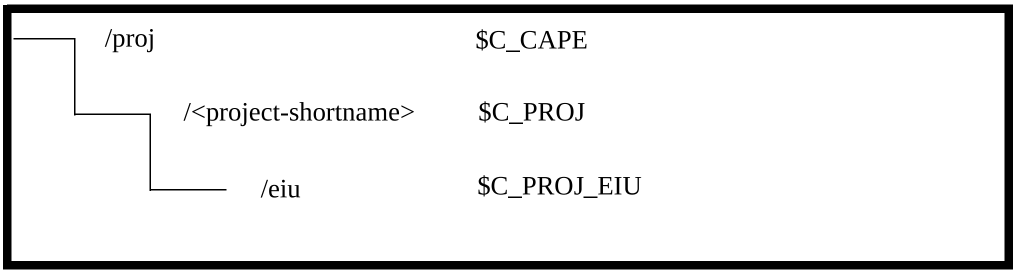


Figure 3-142. Directory Structure

EIU requires that the environment variable be defined. This is done automatically if EIU is started from the Template Builder. If you start EIU from the command line, you have to define the environment variable manually with the following command:

```
export C_PROJ_EIU=<directory specification>
```

The following input files of EIU must be placed in the directory:

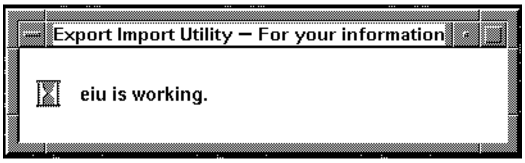
- the controlfile (for import only)
- the selectfile
- the datafile (for import only).

The following output files reside in the directory:

- the logfile
- the datafile (from export only).

User Interface

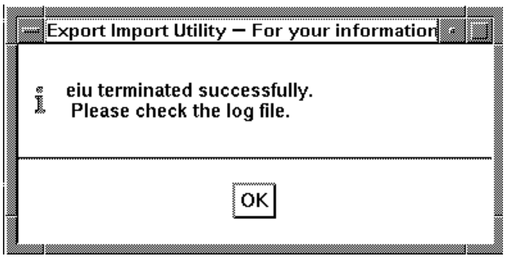
The EIU can be started from the command line with parameters or via the **Import** or **Export** menu item in the Template Builder **File** Menu. Refer to Section 3.4.9.6, Examples (based on the project MAS_EXP) and to Section 3.5.11.1, File Menu for instructions on how to use the Template Builder to start the EIU. After EIU has been started, a message is displayed to indicate the EIU is working, Figure 3-143.



eiu is working.

Figure 3-143. EIU Working Message

This message remains until EIU has finished, at which time another message is displayed to indicate success or failure, Figure 3-144. The second message can be removed by pressing the **OK** button.



eiu terminated successfully.
Please check the log file.

Figure 3-144. EIU Termination Message

Please check the log file both if eiu terminated with errors and if eiu terminated successfully, as SQL*Loader errors and some data errors are not reported within this window.

All message windows can be suppressed by using the command line option *quiet*.

3.4.9.4 Import

The import function is used to import data from a file into the AdvaBuild database. This data must comply with certain format requirements and the format of the data file must be described in a control file. The data is stored in the data file or in files, which are to be stored in the database as file objects. Each line of the data file represents one object or link to be inserted into the database. The data can be in a fixed format (each attribute has a certain position in the line), or in a delimited format (attributes are separated by a special character). This format is described in the control file. The data is loaded into a temporary table by the SQL*Loader first, and then it is inserted into the database. The control file is used as a loader description and as a description of the rows of the temporary table which is created in the database and removed after the insertion of data into the database. An optional selectfile may be specified which describes how the data from the temporary table is loaded into the target tables of the database.

Command Line Arguments

If EIU is started from the command line, the parameters have to be defined within the command line. Its syntax is:

```
eiu import [quiet] uid=user_id pwd=pass_word \
ctl=contol_file dat=data_file [sel=select_file] \
[cat=category] [poi=parent_object_id] [oty=object_type] \
[lty=link_type] [tsk=task]
```

where the back-slash is used to indicate that the command line continues in the next line.

The first parameter following the command eiu, import, is necessary to activate the import function.

Most further parameters are provided in the form

xxx=parameter

where **xxx** is the parameter name and **parameter** is the value. If the parameter uses special symbols like '=', the parameter should be provided in double quotation marks.

Valid parameters are described in Table 3-12. The poi, oty and lty parameters must have single quotation marks (for example, poi='#err'). Some parameters are optional. Others are required.

Table 3-12. Command Line Parameters Import

Parameter	Description
import	activate import function
quiet	suppress working and termination windows (for batch program execution; optional)
uid=user_id	user_id of the database
pwd=pass_word	pass_word for database login
ctl=control_file	name of the control file
dat=data_file	name of the data file (the log file name is created as data_file.log)
sel=select_file	name of the select file (necessary to import STA or FIL objects)
cat=category	category of data to insert. Possible values are: - objects - links (optional; default is objects)
poi=parent_object_id	name of the parent object ID of objects to insert (only if category is objects; optional)
oty=object_type	type of objects to insert (only if category is objects; optional if specified in the select file)
lty=link_type	type of links to insert (only if category is links; optional if specified in the select file)
tsk= task	task to be performed. Possible tasks are: INSERT: links or objects are only inserted, existing links or objects are not updated UPDATE: links or objects are inserted; if they exist they are updated (i.e. the type specific attributes are updated). CLOSE: open links are closed and updated; if they do not exist they are not inserted (parameter is optional if a select file exists)

If parent_object_id (poi), object_type (oty) and link_type (lty) are not specified, they are taken from the input data (where c_p_tree_id is supplied instead of parent_object_id) or from the select file.

Command line parameters always have the lowest priority; that is to say, if an object_type, link_type or parent_object_id is supplied both on the command line and in the input data or in the select file, it is taken from the input data or from the select file.

The files are expected in the directory specified by the environment variable C_PROJ_EIU. All filenames must be provided with an extension, for example, *test.sel*. Illegal or missing parameters result in program interruption and creation of the log file eiu.log in the default directory. This file describes only the first error detected.

The Control File

The control file describes the format of the input data file and the structure of the temporary table. It is therefore needed to load the data file into the temporary table.

The control file syntax is a genuine subset of the ORACLE SQL*Loader control file syntax. The full syntax is:

LOAD [DATA]

[CHARACTERSET character_set_name]

[APPEND | REPLACE | INSERT]

[CONCATENATE {n | (n)} | CONTINUEIF { [THIS | NEXT] [(] (start[:end]) operator {'char_string' | X'hex_string'} [)] | LAST operator {'char_string' | X'hex_string'} }]

INTO TABLE tablename

[FIELDS [delimiter_spec]]

[TRAILING [NULLCOLS]]

[SKIP n]

(column_name {RECNUM | [POSITION ({start [:end] | * [+n]})] datatype_spec}

[,...])

Table 3-13 briefly describes the above lines. For details refer to *Oracle7 Server Utilities Users Guide*.

Table 3-13. Control File Syntax

Control file statement	Description
LOAD [DATA]	Standard first line of control file
[CHARACTERSET charactersetname]	Name of a special characterset (optional; normally not needed)
[APPEND | REPLACE | INSERT]	Append, replace or insert data into the temporary table (as the temporary table is always created new, not necessary)
[CONCATENATE ...	Command to concatenate several data file rows to one database table row (for details refer to *ORACLE handbook*; optional)
INTO TABLE tablename	Name of the temporary table
[FIELDS [delimiter_spec]]	Specification of the delimiter between the columns of the input data file (for syntax of delimiter_spec see table below)
[TRAILING [NULLCOLS]]	Clause to tell the SQL*Loader to treat any relatively positioned columns that are not present in the record as null columns when loading them into the temporary table (optional; normally not needed)
[SKIP *n*]	Skip the first *n* rows in the input data file (optional; normally not needed)
column_name	Name of the column of the temporary table
RECNUM	Record number within the data file as value of the column

Table 3-13. Control File Syntax (Continued)

Control file statement	Description
[POSITION ({ start [:end] | * [+n] })]	Position of each column within the data file lines, if data are provided in a fixed format, i.e. each database column is loaded from a specific column within the data file lines
datatype_spec	Specification of the datatype of the column

Valid datatype specifications (datatype_spec) are:

INTEGER [EXTERNAL [(length)] [delimiter_spec]]

FLOAT [EXTERNAL [(length)] [delimiter_spec]]

ZONED { (precision [, scale]) | EXTERNAL [(length)] [delimiter_spec] }

DECIMAL { (precision [, scale]) | EXTERNAL [(length)] [delimiter_spec] }

DOUBLE

SMALLINT

BYTEINT

VARCHAR (maximum_length)

CHAR [(length)] [delimiter_spec]

DATE [(length)] ["mask"] [delimiter_spec].

Table 3-14 provides a brief description of those datatypes:

Table 3-14. Control file datatypes

Datatype	Short description
INTEGER	Full word binary coded integer
INTEGER EXTERNAL [(length)] [delimiter_spec]	Full word integer written in character form (length is the number of digits, default value is 38 digits)
FLOAT	Binary coded floating point number
FLOAT EXTERNAL [(length)] [delimiter_spec]	Floating point number in character form (length is the maximum length)
ZONED (precision [, scale])	Data in zoned decimal format, i.e. a string of decimal digits, one per byte, with the sign in the last byte (precision: number of digits in the number; scale: number of digits to the right of the decimal point)
ZONED EXTERNAL [(length)] [delimiter_spec]	Zoned data in character form (length is the maximum length)
DECIMAL (precision [,scale])	Decimal data in packed decimal format, with two digits per byte, except for the last byte, which contains a digit and a sign (precision: number of digits in the value; scale: number of digits to the right of the decimal point, default is zero, i.e. a integer number)

Table 3-14. Control file datatypes (Continued)

Datatype	Short description
DECIMAL EXTERNAL [(length)] [delimiter_spec]	Character coded form of decimal number (length is the maximum length)
DOUBLE	Double-precision floating point binary number
SMALLINT	Decimal value of a half word binary integer
BYTEINT	Decimal value of the binary representation of one byte
VARCHAR (maximum_length)	Binary coded varchar, including the length field (maximum_length denotes the maximum character length of the varchar string)
CHAR [(length)] [delimiter_spec]	Character data, the most common data type (length is the maximum string length, if it is omitted a length of 1, i.e. one single character, is the default value)
DATE [(length)] ["mask"] [delimiter_spec]	Date in character form, converted to an ORACLE date using the specified date mask

In contrast to the standard use of control files for the SQL*Loader, the length of a datatype specification is also used by EIU to create the temporary table and therefore should not be omitted, even if the length is also described otherwise (for example, by a POSITION statement). EXTERNAL datatypes are used to import ASCII-coded numerical data. Datatypes without EXTERNAL specification are used to import binary coded numerical data. As there is an additional datatype conversion when importing data into the AS500ES database, you can usually use the datatype CHAR, even for character coded numerical data.

Valid delimiter specifications (delimiter_spec) are:

> [TERMINATED [BY] {WHITESPACE | char | X hex_char} [[OPTIONALLY]
> ENCLOSED [BY] { char | X hex_char } [AND { char | X hex_char }]] |
> ENCLOSED [BY] { char | X hex_char } [AND { char | X hex_char }]]

where char can be enclosed in single or double quotation marks. The character '(' is NOT allowed as a delimiter or enclose character. You must check that a table with the tablename specified within the control file does not exist in the data base. Otherwise the SQL*Loader cannot create the table and the import function will provide an ORACLE error message, if EIU does not work in the quiet mode along with an offer to drop the existing table. Table 3-15 describes the syntax of the delimiter specifications.

Table 3-15. Delimiter specifications

Delimiter specification	Short description
TERMINATED [BY] {WHITESPACE | char | X hex_char}	Clause denoting the delimiter between two data fields: WHITESPACE (i.e. space, tabulator or carriage return) or another character, in single quotation marks, double quotation marks (if it is a single quotation mark) or as a hexadecimal code preceded by an 'X')

Table 3-15. Delimiter specifications (Continued)

Delimiter specification	Short description
[OTIONALLY] ENCLOSED [BY] {char | X hex_char}	Enclosure of data fields with characters (no white space), optional if OPTIONALLY is chosen
AND {char | X hex_char}	Enclosure is by two different characters if the AND clause is used

For further information on the control file see *ORACLE7 Server Utilities User's Guide.*

Restrictions in comparison to the original control file syntax are described in Section 5.2, Export Import Utility Restrictions

The Data File

The data file must contain the data in the form described by the control file. This format may be:

delimited	Every field is separated, for example, by a comma, as specified in the delimiter specification in the control file.
OR	
fixed	Every field begins and ends at certain column positions in a line, as specified in the POSITION clause in the control file.

NOTE

- Import data must not contain the reserved character strings <!> and <!!>.
- INTEGER EXTERNAL columns must contain data if they are to be imported by the SQL*Loader, whereas CHAR columns may be empty. To create null values in a NUMBER attribute in the database, specify the column as CHAR column with an appropriate length in the control file. The SQL*Loader will load data file rows with empty CHAR columns into temporary table rows with null columns. Afterwards eiu will perform type conversion automatically.

The Select File

The select file is an optional file that contains names of attributes and the where-condition for select statements from the temporary table. If there is no select file, all attributes are selected from the temporary table without any where-conditions and they are inserted into the target table with the same attribute names as in the temporary table. If a select file name is specified, the names of the attributes to select and to insert are read from this file.

Syntax of the select file:

```
CATEGORY category
[GROUP group]
TASK task
[PARENT parent_object_id]
SELECT [ALL | DISTINCT]
ATTRIBUTE    target_attribute      select_attribute
[ATTRIBUTE...]
```

FROM tables

WHERE where_condition
:
:

[CATEGORY...] --> begin of the second select block etc.

This syntax is described in Table 3-16.

Table 3-16. Import select file syntax

Select file line	Description
CATEGORY category	The category is either OBJECTS or LINKS. This denotes whether objects or links are to be inserted into the database. It overrides the command line parameter category.
[GROUP group]	The group may be LNK for category LINKS, or OBJ, STA, or FIL for category OBJECTS. A group specification is optional. The default group is OBJ for objects and LNK for links. STA objects and FIL objects can only be imported by use of a select file.
TASK task	The task is either INSERT, UPDATE or CLOSE. If task is INSERT, objects or links (as specified by category) will be inserted into the database, if they do not exist yet, i.e. if there is no object with the same obj_id or no link with the same link_type, which links the same two objects (several identical open links are possible). File objects are inserted, if they do not exist yet, and the files are stored into the database. If a file object already exists, the file is not stored (in this case use UPDATE). If task is UPDATE, objects or links will be inserted into the database, if they do not exist yet, as with INSERT. If they already exist, their type specific attributes will be updated. File objects are also updated, and files are stored into the database or replaced, if they already exist. If task is CLOSE, which is only possible with category links, links which already exist as open links, will be closed and their type specific attributes will be updated. This select file parameter overrides the task given in on the command line.
[PARENT parent_object_id]	The optional parameter parent overrides the poi parameter in the command line. It is valid only for one select block and it is overridden by parent tree ID's read from the temporary table as attributes (for example, ATTRIBUTE c_p_tree_id temporary_table_attribute (only for category objects))
SELECT [ALL | DISTINCT]	The select statement to get data from the temporary table and possibly further tables to insert into the database is built from the information received from the select file after the SELECT keyword. If one of the optional parameters DISTINCT or ALL is specified, SELECT DISTINCT or SELECT ALL is executed.
ATTRIBUTE target_attribute select_attribute	The target_attribute is the name of the target attribute to insert into the database table. It can differ from select_attribute, which is the name of the select attribute to read from the temporary table containing the attributes described in the control file or from other tables described in the FROM statement. Select_attribute may also contain concatenations, the SUBSTR function and all other valid SQL SELECT's.

Table 3-16. Import select file syntax (Continued)

Select file line	Description
FROM tables	This specifies the tables to select data from. Usually the temporary table specified in the control file is one of those tables. If there are several tables, their names are separated by commas.
WHERE where_condition	where_condition is a where condition for the SELECT statement from the tables specified by FROM. It may contain subqueries, unions, GROUP BY statements and so on, which can follow the WHERE keyword in a SQL SELECT-statement. The where condition itself is optional, but the keyword WHERE may not be omitted.

This will create a select statement such as the one shown below:

SELECT [ALL | DISTINCT] select_attribute, select_attribute,...

FROM tables WHERE where_condition

Data for several selects using the same temporary table can be stored in one select file which consists of several select blocks.

There are certain and **necessary target attributes** in the select statement:

For category OBJECTS and group OBJ or FIL:

- obj_id
- obj_type or the command line parameter oty (if the attribute obj_type is specified, the command line parameter oty is not used)
- c_p_tree_id or the PARENT line or the command line parameter poi.

For category OBJECTS and group FIL:

- file_name (the name of the file in the file system to be stored, i.e. the physical filename on the disc)
- path (the path where to find the file; if it is not specified, the file is expected to be found in the directory specified by the environment variable C_PROJ_EIU).

For category OBJECTS and group STA:

- obj_type or the command line parameter oty (this specifies type STA object type; therefore obj_type may not be an attribute name of an STA object, which is to be imported by EIU).

For category LINKS:

- obj_id1
- obj_id2
- link_type or the command line parameter lty (if the attribute link_type is specified, the command line parameter lty is not used).

In category LINKS and task CLOSE:

- c_int_key.

Objects are inserted into the AdvaBuild database as list objects by default (they are not shown in the tree structure). If an object should be shown in the tree structure, the attribute c_disp_char has to be set to 'T' (for example, ATTRIBUTE c_disp_char 'T').

If you specify an attribute more than once, it is uncertain which specification is used. In some cases an error may occur. Specifications of select attributes, where condition and others may extend to the next or further lines of the select file, until the next keyword or end of file is found.

A line starting with a double dash (--) is interpreted as a comment line and therefore skipped.

The Log Files

Two different log files are created by EIU:

- The log file eiu.log

  This log file is filled if any of the following occur:

  - environment variable C_PROJ_EIU is missing
  - command line parameters are missing or wrong
  - connection to the database fails
  - the other log file cannot be opened
  - the ORACLE SQL*Loader detects fatal errors.

  The log file eiu.log is generated in the default directory that EIU was started from. Only the first error detected is written to eiu.log, then EIU ends immediately. The log file eiu.log is created in any case, but normally empty.

- The standard log file

  The normal log file has the same name as the data file with the additional extension ".log" and is created in the C_PROJ_EIU directory. It contains messages from the SQL*Loader (see SQL*Loader description) and from EIU itself, including data sets which could not be inserted by EIU. Data sets rejected by the SQL*Loader can be found in the bad file (filename like data file name with the extension ".bad", in the same directory). Fatal errors of EIU cause an immediate stop of program execution. Errors concerning single rows only do not cause an immediate stop.

Return Code of the Program

If the program terminates without error, the return code is 0; otherwise the return code is the error code of the EIU error or 200 for non-EIU errors (e.g. database or operation system errors).

Error and Exception Handling

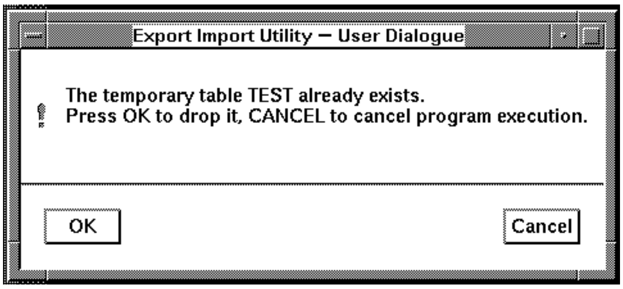
If EIU is ready, a message indicates whether EIU was successful (though there might be errors in single rows, indicated in the log file) or stopped with a fatal error (in this case consult the log files for the message, e.g. "no data found" could be a possible error that the given parent of one object doesn't exist).

In case of fatal errors in the log file eiu.log in the standard directory, check the command line input parameters import, uid, pwd and dat and the environment variable C_PROJ_EIU.

In case of fatal errors in the log file, check the other command line parameters and the syntax of your select file and control file.

In case of errors concerning special rows, check your data file.

If the process is interrupted irregularly (for example, because of a system crash), the temporary table defined in the control file is not deleted. If you start eiu not in the quiet mode and with the same temporary table name the next time, the following window is displayed, Figure 3-144 (the example is for a table named TEST).



The temporary table TEST already exists.
Press OK to drop it, CANCEL to cancel program execution.

Figure 3-145. EIU Drop temporary table dialog

If you press **OK**, the old table is deleted and eiu goes on working. If you press **CANCEL**, the table remains unchanged and eiu terminates with a fatal error.

CAUTION

> There is no check for the table you drop, when using the user dialog shown in Figure 3-144. If you drop a necessary table of the database, you will destroy your database, so please be careful. Make sure, that the table which you want to drop is not a required temporary table.

Error messages from EIU are described in Section 5.1, Error Messages.

3.4.9.5 Export

The export utility exports data sets into a file. It SELECTs rows from the specified database tables and writes them into the file. It also extracts data from file objects to files.

Command Line Arguments

If EIU is started as stand-alone, the parameters are given in the command line. Its syntax is:

```
eiu export [quiet] [header] uid=user_id pwd=pass_word \
dat=data_file [sel=select_file] [tbl=table_name] \
[atn=attribute_names] [wre=where_string] [fmt=format] \
[ctl=control_file] [eop=extract_option] [select_for_update]
```

where the back-slash is used to indicate that the command line continues in the next line.

The first parameter following the command eiu, export, is necessary to activate the export function.

Most further parameters are provided in the form

xxx=parameter

where **xxx** is the parameter name and **parameter** is the value. If the parameter uses special symbols like '=', the parameter should be provided in double quotation marks.

Some parameters are optional. Others are required. All parameters are described in Table 3-17:

Table 3-17. Command Line Parameters Export

Parameter	Description
export	activate export function
quiet	suppress working and termination windows (for batch program execution; optional)
header	create a header for the export data file, containing the select file name (if any), the date and time of the export, the names of the attributes exported from the database, the database tables where data was exported from and the where condition (optional; without effect when extracting files)
uid=user_id	user_id for database login
pwd=pass_word	pass_word for database login
sel=select_file	name of the select file (optional; necessary to extract files)
dat=data_file	name of the data file as target of the export function or as input data file for extract (the log file name is formed as data_file.log)
tbl=table_name	name of the tables to export data from (optional); only necessary if no select file is specified, otherwise ignored; not used for extract
atn=attribute_names	string with all attributes to export, separated by commas (optional); only necessary if no select file is specified, otherwise ignored; not used for extract
wre=where_string	where condition of select statement (optional); only necessary if no select file is specified, otherwise ignored; not used for extract
fmt=format	format of the data file; possible values are: - DELIMITED (data delimited with commas) - FIXED (data at fixed columns in the file) (optional); default value is delimited, not used for extract
ctl=control_file	name of the control file; only for extract
eop=extract_option	extract option to extract a file; possible options are: - C : create file, if it does not exist yet - R : replace file, if it already exists, don't create a new file - CR : create or replace file - RO : replace file with some optimization - CRO : create or replace file with some optimization (optional); default is CR; only used for extract
select_for_update	select files for update (files cannot be modified until given free again); (optional); only for extract

If table name (tbl), attributes (atn) and where condition (wre) are not specified for export, they are taken from the select file. If they are defined and a select file exists, they are taken from the select file which has a higher priority.

The files are expected in the directory specified by the environment variable C_PROJ_EIU. All filenames must be defined with a file extension, for example, *test.sel*.

The Control File

A control file is only needed for extract. It has the same syntax and usage as the import control file.

The Data File

The export data file is filled by EIU with the data selected from the database. Optional header data are also written into the data file if they are selected by the command line parameter header.

The extract data file contains input data for the extract function.

The Select File

The three command line parameters tbl, atn and wre may also be defined in a select file, which overrides command line parameters. Furthermore a select file is necessary to extract files. The structure of this file is:

```
[TASK task]
SELECT [DISTINCT | ALL]
ATTRIBUTE   target_attribute    select_attribute
[ATTRIBUTE...]
FROM tables
WHERE [where_condition]
```

This syntax is described in Table 3-18.

Table 3-18. Export select file syntax

Select file line	Description
TASK task	Task is the task to be performed. Possible tasks are EXPORT to export data from the database (default value) or EXTRACT to extract files from the database.
SELECT [ALL | DISTINCT]	The select statement to select data from the database tables for export or from the temporary table for extract. It is built based on the select file contents after the SELECT keyword. If one of the optional parameters DISTINCT or ALL is specified, SELECT DISTINCT or SELECT ALL is executed.
ATTRIBUTE target_attribute select_attribute	**target_attribute** is not used for export, but may not be omitted (otherwise select_attribute is interpreted as target_attribute and select_attribute is not found); for extract, target_attribute specifies the input data for the function. Those attributes are: file_name : the name of the target file path : the path where to place this file (if it is not specified, the default path is the directory defined by the environment variable C_PROJ_EIU) obj_id : obj_id of the file object to extract from the database **select_attribute** is the name of the attribute to select from the table (concatenations, SUBSTR function and all other valid SQL SELECTS's are allowed).

Table 3-18. Export select file syntax (Continued)

Select file line	Description
FROM tables	**tables** specifies the tables to select data from. If there are several tables, their names are separated by commas.
WHERE where_condition	where_condition is a where condition for the SELECT statement from the tables specified by FROM. It may contain subqueries, unions, GROUP BY statements and so on, which can follow the WHERE keyword in a SQL SELECT-statement. The where condition itself is optional, but the keyword WHERE may not be omitted.

The contents of the select file will result in a select statement such as the one shown below:

```
SELECT [ALL | DISTINCT] select_attribute, select_attribute,...
FROM tables
WHERE where_condition
```

The export select file may only contain data for one select statement (whereas the import select file can contain data for several selects).

Specifications of select attributes, where condition and others may extend to the next or further lines of the select file, until the next keyword or end of file is found.

A line starting with a double dash (--) is interpreted as a comment line and therefore skipped.

The Extract Files

The Extract Files are DRAWINGS and TEXT files, which are extracted from the database using the EIU and the certain Task EXTRACT.

The Log Files

Two different log files are created by EIU:

- The log file eiu.log

  This log file is filled if any of the following occur:

  - the environment variable C_PROJ_EIU is missing
  - command line parameters are missing or obviously wrong
  - connection to the database fails
  - the standard log file cannot be created.

  The log file eiu.log is generated in the default directory that EIU was started from. Only the first error detected is written to eiu.log, then EIU ends immediately. In contrast to IMPORT, eiu.log is only generated if errors occur.

- The standard log file

  The standard log file is named like the data file with the additional extension ".log".
  It contains messages from EIU. Every error immediately stops program execution.

Return Code of the Program

If the program terminates without error, the return code is 0; otherwise the return code is the error code of the EIU error or 200 for non-EIU errors (e.g. database or operation system errors).

Error and Exception Handling

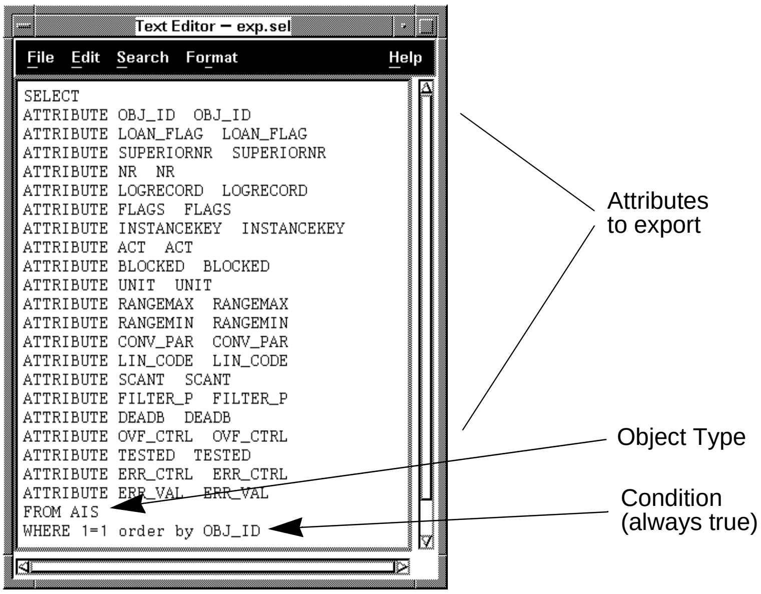
In case of fatal errors announced in the log file eiu.log, check the command line input parameters export, uid, pwd and dat and the environment variable C_PROJ_EIU.

In case of errors in the log file, check the other command line parameters and the syntax of your select file.

3.4.9.6 Examples (based on the project MAS_EXP)

EIU Export

The select file *exp.sel* exports information from all objects of the type AIS.



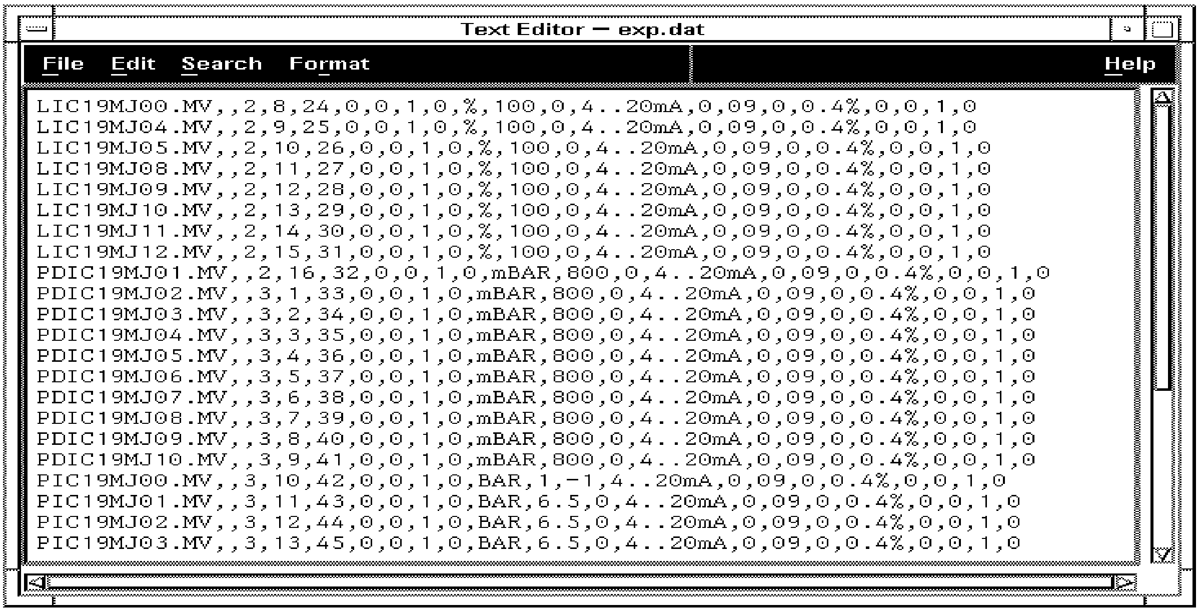
```
SELECT
ATTRIBUTE OBJ_ID  OBJ_ID
ATTRIBUTE LOAN_FLAG  LOAN_FLAG
ATTRIBUTE SUPERIORNR  SUPERIORNR
ATTRIBUTE NR  NR
ATTRIBUTE LOGRECORD  LOGRECORD
ATTRIBUTE FLAGS  FLAGS
ATTRIBUTE INSTANCEKEY  INSTANCEKEY
ATTRIBUTE ACT  ACT
ATTRIBUTE BLOCKED  BLOCKED
ATTRIBUTE UNIT  UNIT
ATTRIBUTE RANGEMAX  RANGEMAX
ATTRIBUTE RANGEMIN  RANGEMIN
ATTRIBUTE CONV_PAR  CONV_PAR
ATTRIBUTE LIN_CODE  LIN_CODE
ATTRIBUTE SCANT  SCANT
ATTRIBUTE FILTER_P  FILTER_P
ATTRIBUTE DEADB  DEADB
ATTRIBUTE OVF_CTRL  OVF_CTRL
ATTRIBUTE TESTED  TESTED
ATTRIBUTE ERR_CTRL  ERR_CTRL
ATTRIBUTE ERR_VAL  ERR_VAL
FROM AIS
WHERE 1=1 order by OBJ_ID
```

Figure 3-147. Select File exp.sel

Do not forget the double parameters for the attributes. The condition "1=1" causes the EIU to export the attributes of all AIS objects. The export is started with the following command lines:

```
>cd $C_CAPE/mas_exp/eiu
>export C_PROJ_EIU="$C_CAPE/mas_exp/eiu"
>eiu export uid=mas_exp_cfg pwd=mas_exp_cfg sel=exp.sel
dat=exp.dat fmt=delimited
```

The resulting data file *exp.dat*: is shown in Figure 3-148,



```
LIC19MJ00.MV,,2,8,24,0,0,1,0,%,100,0,4..20mA,0,09,0,0.4%,0,0,1,0
LIC19MJ04.MV,,2,9,25,0,0,1,0,%,100,0,4..20mA,0,09,0,0.4%,0,0,1,0
LIC19MJ05.MV,,2,10,26,0,0,1,0,%,100,0,4..20mA,0,09,0,0.4%,0,0,1,0
LIC19MJ08.MV,,2,11,27,0,0,1,0,%,100,0,4..20mA,0,09,0,0.4%,0,0,1,0
LIC19MJ09.MV,,2,12,28,0,0,1,0,%,100,0,4..20mA,0,09,0,0.4%,0,0,1,0
LIC19MJ10.MV,,2,13,29,0,0,1,0,%,100,0,4..20mA,0,09,0,0.4%,0,0,1,0
LIC19MJ11.MV,,2,14,30,0,0,1,0,%,100,0,4..20mA,0,09,0,0.4%,0,0,1,0
LIC19MJ12.MV,,2,15,31,0,0,1,0,%,100,0,4..20mA,0,09,0,0.4%,0,0,1,0
PDIC19MJ01.MV,,2,16,32,0,0,1,0,mBAR,800,0,4..20mA,0,09,0,0.4%,0,0,1,0
PDIC19MJ02.MV,,3,1,33,0,0,1,0,mBAR,800,0,4..20mA,0,09,0,0.4%,0,0,1,0
PDIC19MJ03.MV,,3,2,34,0,0,1,0,mBAR,800,0,4..20mA,0,09,0,0.4%,0,0,1,0
PDIC19MJ04.MV,,3,3,35,0,0,1,0,mBAR,800,0,4..20mA,0,09,0,0.4%,0,0,1,0
PDIC19MJ05.MV,,3,4,36,0,0,1,0,mBAR,800,0,4..20mA,0,09,0,0.4%,0,0,1,0
PDIC19MJ06.MV,,3,5,37,0,0,1,0,mBAR,800,0,4..20mA,0,09,0,0.4%,0,0,1,0
PDIC19MJ07.MV,,3,6,38,0,0,1,0,mBAR,800,0,4..20mA,0,09,0,0.4%,0,0,1,0
PDIC19MJ08.MV,,3,7,39,0,0,1,0,mBAR,800,0,4..20mA,0,09,0,0.4%,0,0,1,0
PDIC19MJ09.MV,,3,8,40,0,0,1,0,mBAR,800,0,4..20mA,0,09,0,0.4%,0,0,1,0
PDIC19MJ10.MV,,3,9,41,0,0,1,0,mBAR,800,0,4..20mA,0,09,0,0.4%,0,0,1,0
PIC19MJ00.MV,,3,10,42,0,0,1,0,BAR,1,-1,4..20mA,0,09,0,0.4%,0,0,1,0
PIC19MJ01.MV,,3,11,43,0,0,1,0,BAR,6.5,0,4..20mA,0,09,0,0.4%,0,0,1,0
PIC19MJ02.MV,,3,12,44,0,0,1,0,BAR,6.5,0,4..20mA,0,09,0,0.4%,0,0,1,0
PIC19MJ03.MV,,3,13,45,0,0,1,0,BAR,6.5,0,4..20mA,0,09,0,0.4%,0,0,1,0
```

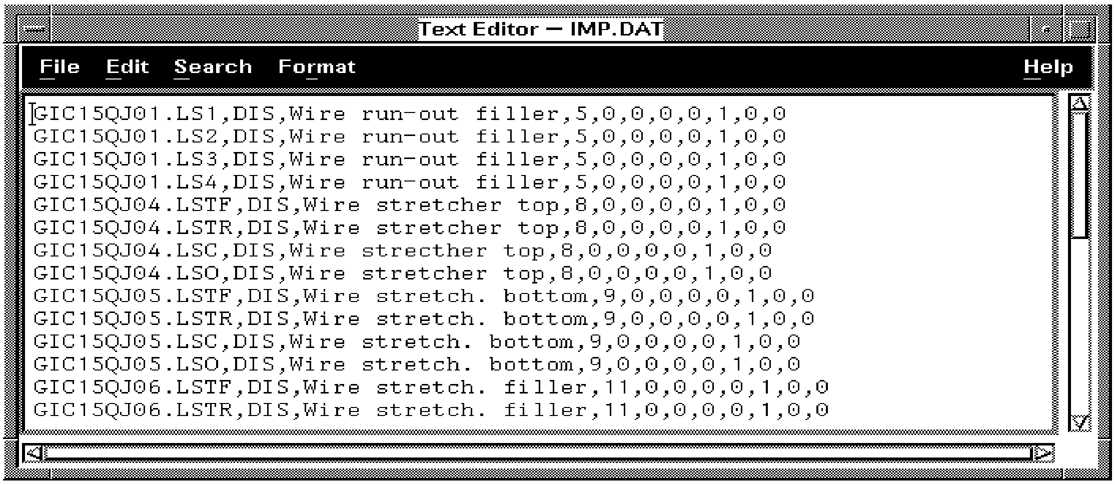
Figure 3-148. Resulting Data File

EIU Import

The files IMP.DAT, IMP.CTL and IMP.SEL import objects to the type DIS under self created TAG objects in the MAS Example Project. You can find them in the directory:

$C_CAPE/mas_exp/eiu

Data file IMP.DAT is shown in Figure 3-149.

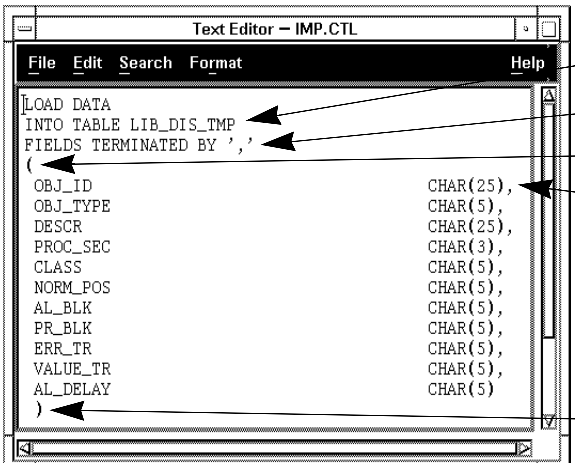


```
GIC15QJ01.LS1,DIS,Wire run-out filler,5,0,0,0,0,1,0,0
GIC15QJ01.LS2,DIS,Wire run-out filler,5,0,0,0,0,1,0,0
GIC15QJ01.LS3,DIS,Wire run-out filler,5,0,0,0,0,1,0,0
GIC15QJ01.LS4,DIS,Wire run-out filler,5,0,0,0,0,1,0,0
GIC15QJ04.LSTF,DIS,Wire stretcher top,8,0,0,0,0,1,0,0
GIC15QJ04.LSTR,DIS,Wire stretcher top,8,0,0,0,0,1,0,0
GIC15QJ04.LSC,DIS,Wire strecther top,8,0,0,0,0,1,0,0
GIC15QJ04.LSO,DIS,Wire stretcher top,8,0,0,0,0,1,0,0
GIC15QJ05.LSTF,DIS,Wire stretch. bottom,9,0,0,0,0,1,0,0
GIC15QJ05.LSTR,DIS,Wire stretch. bottom,9,0,0,0,0,1,0,0
GIC15QJ05.LSC,DIS,Wire stretch. bottom,9,0,0,0,0,1,0,0
GIC15QJ05.LSO,DIS,Wire stretch. bottom,9,0,0,0,0,1,0,0
GIC15QJ06.LSTF,DIS,Wire stretch. filler,11,0,0,0,0,1,0,0
GIC15QJ06.LSTR,DIS,Wire stretch. filler,11,0,0,0,0,1,0,0
```

Format of ASCII data:
OBJ_ID, OBJ_TYPE, DESCR, PROC_SEC, CLASS, NORM_POS ....

Figure 3-149. Data File IMP.DAT

Control file IMP.CTL is shown in Figure 3-150



```
LOAD DATA
INTO TABLE LIB_DIS_TMP
FIELDS TERMINATED BY ’,’
(
 OBJ_ID                                      CHAR(25),
 OBJ_TYPE                                    CHAR(5),
 DESCR                                       CHAR(25),
 PROC_SEC                                    CHAR(3),
 CLASS                                       CHAR(5),
 NORM_POS                                    CHAR(5),
 AL_BLK                                      CHAR(5),
 PR_BLK                                      CHAR(5),
 ERR_TR                                      CHAR(5),
 VALUE_TR                                    CHAR(5),
 AL_DELAY                                    CHAR(5)
 )
```

Name of temporary table

Delimiter is comma

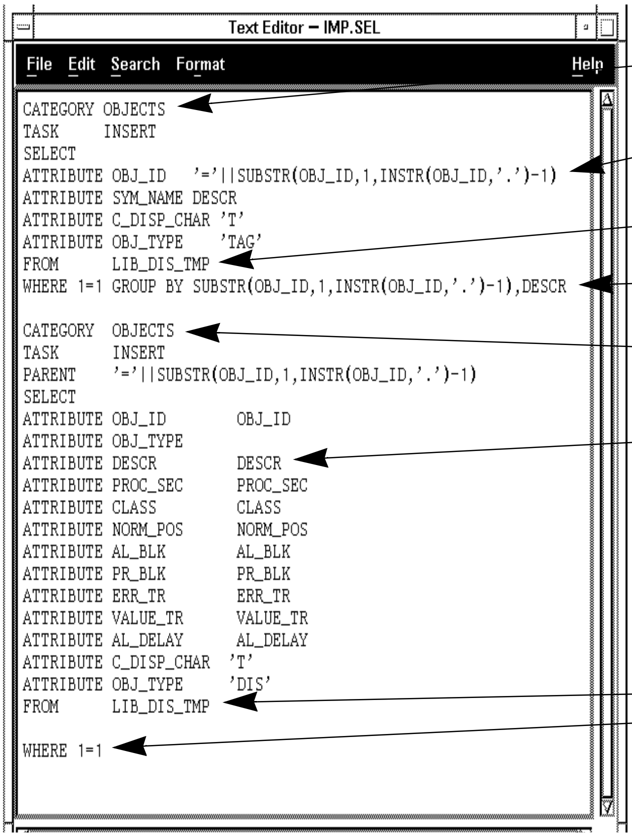
Open field definitions

First field has 25 characters, the field’s name is OBJ_ID

Close field definitions

Figure 3-150. Control File IMP.CTL

Select file IMP.SEL is shown in Figure 3-151.



```
CATEGORY OBJECTS
TASK     INSERT
SELECT
ATTRIBUTE OBJ_ID   ’=’||SUBSTR(OBJ_ID,1,INSTR(OBJ_ID,’.’)-1)
ATTRIBUTE SYM_NAME DESCR
ATTRIBUTE C_DISP_CHAR ’T’
ATTRIBUTE OBJ_TYPE    ’TAG’
FROM      LIB_DIS_TMP
WHERE 1=1 GROUP BY SUBSTR(OBJ_ID,1,INSTR(OBJ_ID,’.’)-1),DESCR

CATEGORY  OBJECTS
TASK      INSERT
PARENT    ’=’||SUBSTR(OBJ_ID,1,INSTR(OBJ_ID,’.’)-1)
SELECT
ATTRIBUTE OBJ_ID        OBJ_ID
ATTRIBUTE OBJ_TYPE
ATTRIBUTE DESCR         DESCR
ATTRIBUTE PROC_SEC      PROC_SEC
ATTRIBUTE CLASS         CLASS
ATTRIBUTE NORM_POS      NORM_POS
ATTRIBUTE AL_BLK        AL_BLK
ATTRIBUTE PR_BLK        PR_BLK
ATTRIBUTE ERR_TR        ERR_TR
ATTRIBUTE VALUE_TR      VALUE_TR
ATTRIBUTE AL_DELAY      AL_DELAY
ATTRIBUTE C_DISP_CHAR  ’T’
ATTRIBUTE OBJ_TYPE     ’DIS’
FROM      LIB_DIS_TMP

WHERE 1=1
```

Create new objects

Create Tag name from DIS OBJ_ID without “.xxx”

Name of temporary table

Condition, only one TAG with same OBJ_ID

Create new objects Parent’s name (TAG)

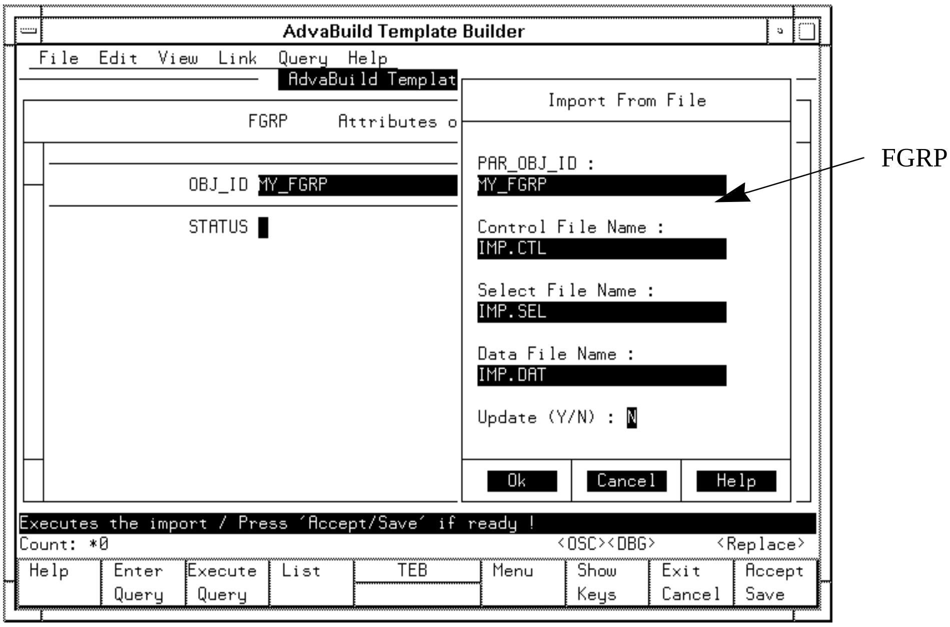
Source is field of temp. table

Name of temp. table
Condition, always true

Figure 3-151. Select File IMP.SEL

Create an Object with object type "FGRP" and start TEB.

Figure 3-152 illustrates the import via Template Builder



OBJ_ID MY_FGRP

STATUS

PAR_OBJ_ID : MY_FGRP

Control File Name : IMP.CTL

Select File Name : IMP.SEL

Data File Name : IMP.DAT

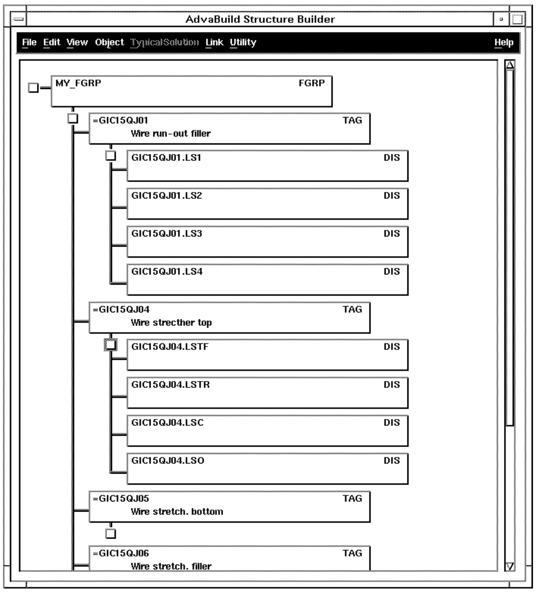
Update (Y/N) : N

Executes the import / Press 'Accept/Save' if ready !

Count: *0 <OSC><DBG> <Replace>

Figure 3-152. Starting Import Via Template Builder

The resulting TAG and DIS objects in the Structure Builder Figure 3-153.



MY_FGRP FGRP
- =GIC15QJ01 TAG Wire run-out filler
  - GIC15QJ01.LS1 DIS
  - GIC15QJ01.LS2 DIS
  - GIC15QJ01.LS3 DIS
  - GIC15QJ01.LS4 DIS
- =GIC15QJ04 TAG Wire strecther top
  - GIC15QJ04.LSTF DIS
  - GIC15QJ04.LSTR DIS
  - GIC15QJ04.LSC DIS
  - GIC15QJ04.LSO DIS
- =GIC15QJ05 TAG Wire stretch. bottom
- =GIC15QJ06 TAG Wire stretch. filler

Figure 3-153. Resulting TAG and DIS Objects

3.4.10 How to Import Existing FrameMaker and AutoCAD Files

This section provides instructions for importing existing FrameMaker and AutoCAD files into TEXT and DRAWING objects. This section includes instructions for handling files on floppy disks.

3.4.10.1 Handling Floppy Disks

In order to import existing FrameMaker and AutoCAD files, you must first know how to read and copy the files on floppy disks. To list the contents of a floppy disk, use the following command:

```
dosls /dev/rdsk/c201d0s0:/<directory>
```

where <*directory*> is a directory or path specification on the floppy disk

To copy a file from the floppy disk to the current directory on the Advant Station, use the following command:

```
doscp /dev/rdsk/c201d0s0:/<directory>/<file1> ./<file2>
```

where <*directory*> is a directory or path specification on the floppy disk, <*file1*> is the name of the source file (being copied), and <*file2*> is the name of the destination file on the Advant Station which is normally the same as <file1>

To copy a file from the current directory on the Advant Station to a floppy disk, use the following command:

```
doscp ./<file1> /dev/rdsk/c201d0s0:/<directory>/<file2>
```

where <*file1*> is the name of the source file (being copied), <*directory*> is a directory or path specification on the floppy disk, and <*file2*> is the name of the destination file on the floppy disk which has to be in the DOS naming rules

3.4.10.2 How to Import an Existing FrameMaker File

To import an existing FrameMaker file from a floppy disk to the Advant Station:

1. Copy the existing file (<*file_name*>**.doc**) from the floppy disk into a directory on the Advant Station.

   See Section 3.4.10.1, Handling Floppy Disks for details. Use the following path specification (<*directory*>):

   /proj/<*project*>**/fil**

   where **<***project***>** is the target project name (for example: mod_exp).

2. Insert a TEXT object in the project structure via the Structure Builder.
3. Open the Template Builder view of the TEXT object and edit the following attribute:

   DOC_TYPE

   The list of values function is available for this object attribute.

4. Open the Document Builder (FrameMaker) view of this TEXT object.
5. Open the copied FrameMaker file using **Open** from the **File** menu in the new created FrameMaker.
6. Save this copied FrameMaker file with the name of the other FrameMaker file (which was opened by STB), displayed in the **Window Title** of them. Use **Save As...** in the **File** menu.
7. Exit both files without saving, and exit FrameMaker.

3.4.10.3 How to Import an Existing AutoCAD file

To import an existing AutoCAD file from a floppy disk to the Advant Station:

1. Copy the existing file (<*file_name*>**.dwg**) from the floppy disk into a directory on the Advant Station.

   Make sure that the existing drawing has no frame or footer.

   See Section 3.4.10.1, Handling Floppy Disks for details. Use the following path specification (<*directory*>):

   /proj/<*project*>**/fil**

   where **<***project***>** is the target project name (for example: mod_exp).

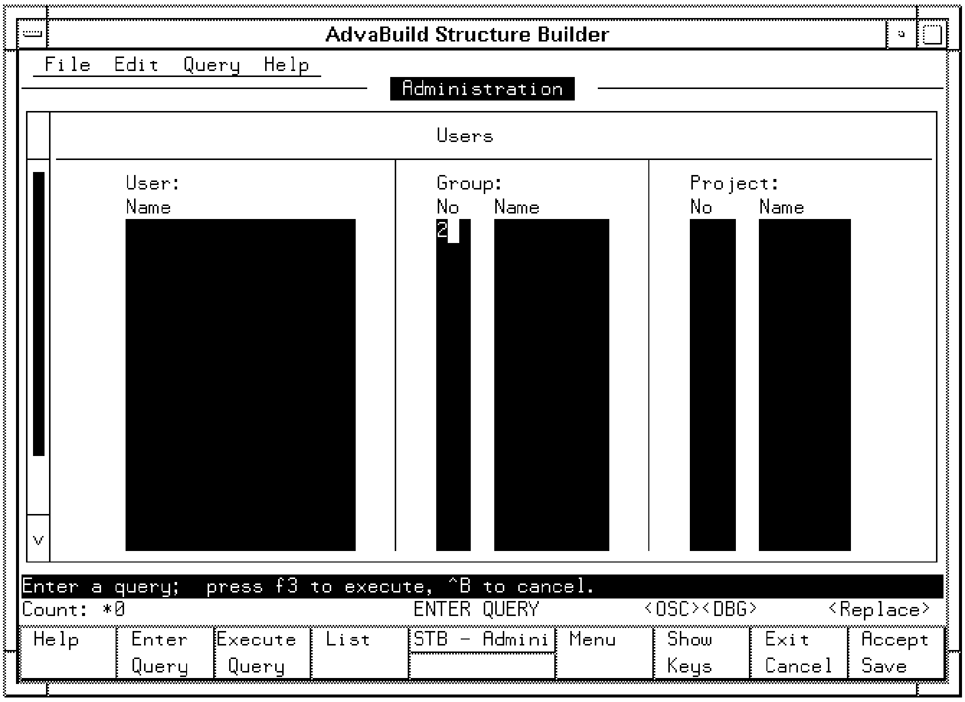
2. Insert a DRAWING object in the project structure via the Structure Builder.
3. Open the Template Builder view of the DRAWING object and edit the attributes.
4. Open the Diagram Builder view of the DRAWING object.
5. Insert the existing drawing into the new DRAWING object using the AutoCAD **ddinsert** command.
6. Save DRAWING by using **qsave** command.
7. Quit Diagram Builder by **quit** command.

3.4.11 Queries

The **Query** function provides the means to define a specific set of criteria to identify and retrieve one or more objects from the AdvaBuild database. The query function is provided in all AdvaBuild applications that use the FORMS user interface. The method for defining and executing the queries is the same for all applications, except for Link Handling which has a special method for generating queries (See Section 3.4.4, Application Procedures for Link Handling for details regarding querying in Link Handling applications). The Query functions are available via function keys on the bottom line and under the **Query** menu.

3.4.11.1 How to Enter a Query

You define a query via **Enter Query**. This clears the current window and places the cursor in the first field. You enter data in one or more fields to define the criteria that the applicable database items are checked against, Figure 3-154.



AdvaBuild Structure Builder

Administration

Users

User:
Name

Group:
No Name
2

Project:
No Name

Enter a query; press f3 to execute, ^B to cancel.

Count: *0 ENTER QUERY <OSC><DBG> <Replace>

Figure 3-154. Example, Enter Query

In this example, the query will retrieve all users whose group number is 2 (CONFIG users). Generally, you enter a specific value in the field based on the range (or set) of valid values for that field. You can also use relational operators to enter a range of values, Table 3-19.

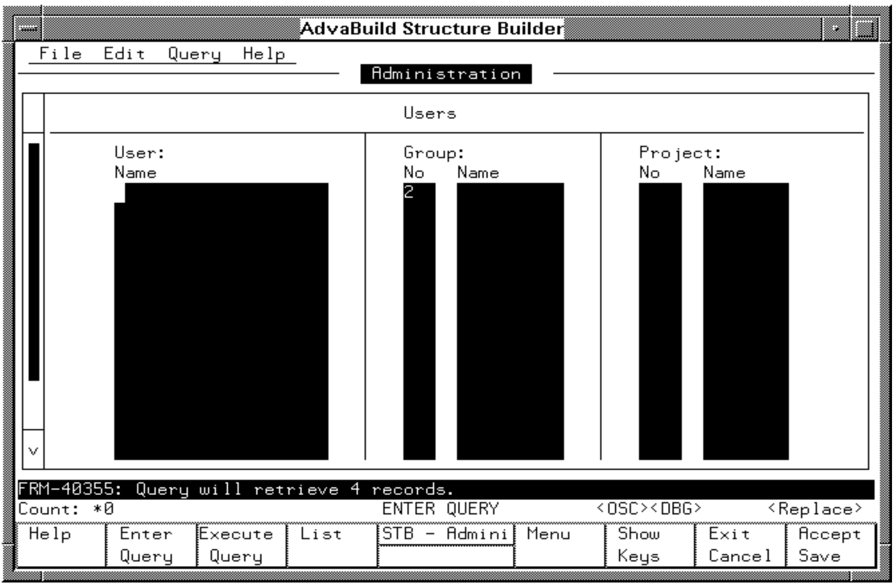
Table 3-19. Relational Operators for Queries

Operator	Meaning	Example
=	equal to	= 'smith'
!=	not equal to	!= 19.5
>	greater than	> 100.00
>=	greater than or equal to	>= 2000
<	less than	< 'smith'
<=	less than or equal to	<= 100.00
BETWEEN	between two values	#BETWEEN 120 AND 360

The length of the statement cannot exceed the size of the field. Character and date values (for example, 'smith') must be enclosed by single quotes. When you use the relational operator BETWEEN, you must precede the operator with a #.

3.4.11.2 Count Hits

After you enter a query and before you execute it, you can use **Count Hits** to determine the number of items that the query will retrieve. The number of "hits" is displayed on the message line. After using **Count Hits**, you can **Execute** the query, **Enter** a different query, or **Cancel** the query. Figure 3-155 shows the result of **Count Hits** for the example in Figure 3-154.



Group No: 2

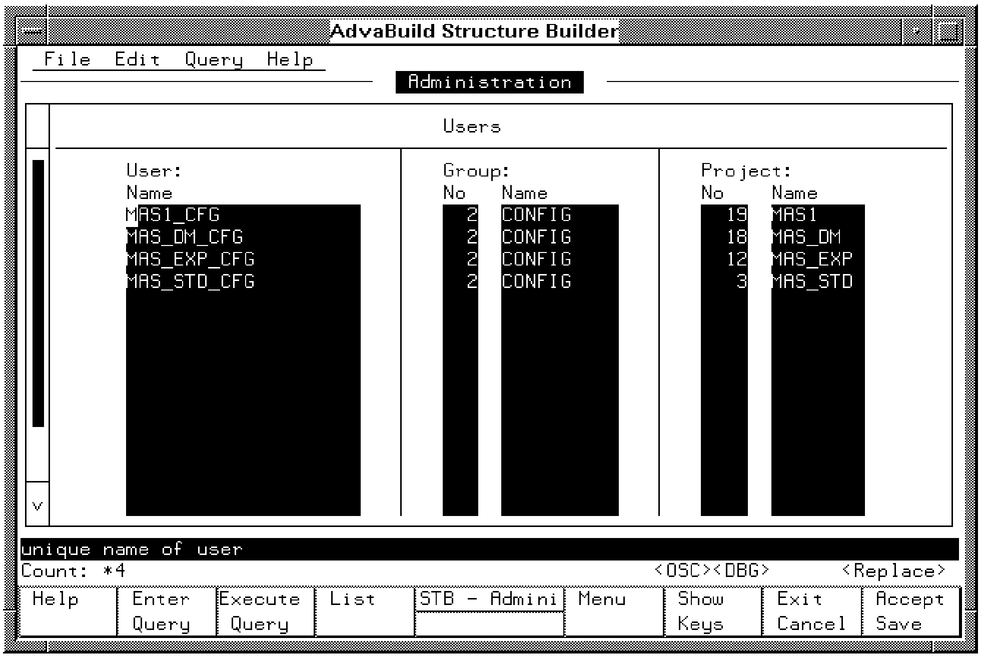
```
FRM-40355: Query will retrieve 4 records.
Count: *0                    ENTER QUERY          <OSC><DBG>        <Replace>
```

In this example four items will be returned

Figure 3-155. Example Count Hits

3.4.11.3 How to Execute a Query

When you execute the query, the items that meet the criteria are retrieved. For windows that only display one item at a time, such as the single-object template form, you can use the up and down arrow keys to scroll through the list of items. Figure 3-156 shows the result of executing the query in the example in Figure 3-154.



User: Name	Group: No	Group: Name	Project: No	Project: Name
MAS1_CFG	2	CONFIG	19	MAS1
MAS_DM_CFG	2	CONFIG	18	MAS_DM
MAS_EXP_CFG	2	CONFIG	12	MAS_EXP
MAS_STD_CFG	2	CONFIG	3	MAS_STD

unique name of user

Count: *4 <OSC><DBG> <Replace>

Figure 3-156. Example, Execute Query

For example, to retrieve all projects of a certain size:

1. Open the Projects window.
2. Select **Enter Query**.
3. Move the cursor to the Pro-Size field and enter the appropriate size.
4. Select the **Execute Query** function.

The system searches the database for all projects of the specified size and lists them in the Projects window.

3.4.11.4 Cancel

Cancel Query cancels the query and restores the original window.

3.5 Menus

This section describes menu items for all AdvaBuild menus on a menu-by-menu and item-by-item basis. In this section, the menus are organized according to the windows where they occur, for instance, Section 3.5.2, Menus for the Administration Window, Section 3.5.5, Menus for Structure Builder Tree and List Presentation Windows, and so on.

Two menus that occur in a number of AdvaBuild applications are **Query** and **Help**. Rather than repeat the descriptions for these menus over and over, they are described just once in Section 3.5.1, Common Menus.

3.5.1 Common Menus

3.5.1.1 Query Menu

The **Query** menu is shown in Figure 3-157.

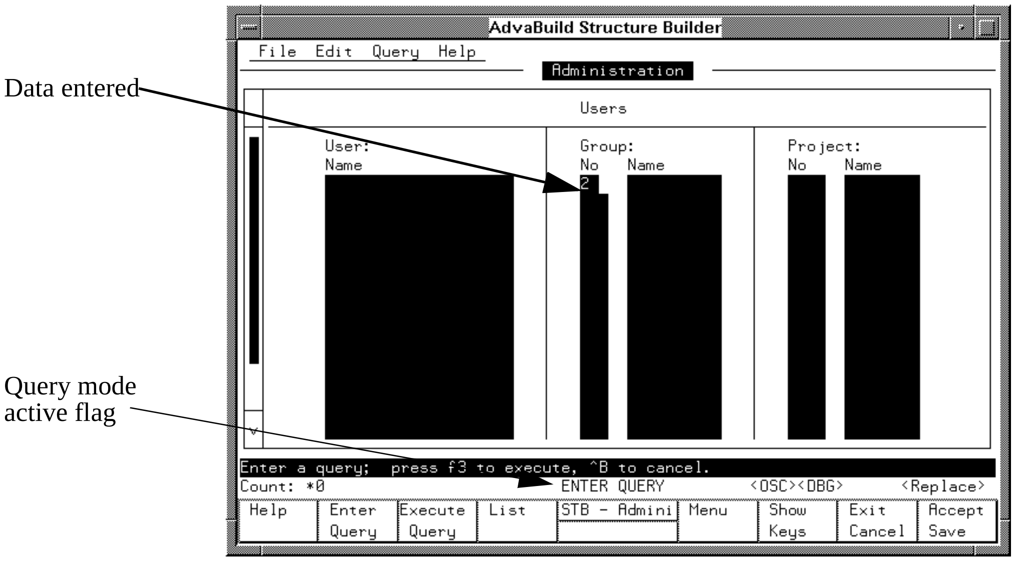
Query

Enter
Count Hits
Execute
Cancel

Figure 3-157. Query Menu

Query - Enter

Enter is used to define a query that searches for data items according to a special search mask. **Enter** clears the data area and places the cursor in the first data field to enter the query, Figure 3-158.



```
Enter a query;  press f3 to execute, ^B to cancel.
Count: *0                    ENTER QUERY        <OSC><DBG>       <Replace>
```

Figure 3-158. Enter Query

The **Query Enter** function in the List presentation of the Structure Builder provides a dialog box for specifying the query parameters, Figure 3-159.

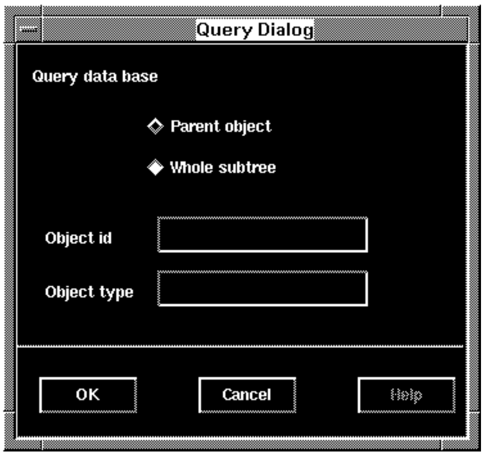


Figure 3-159. Query Enter Function for List Presentation

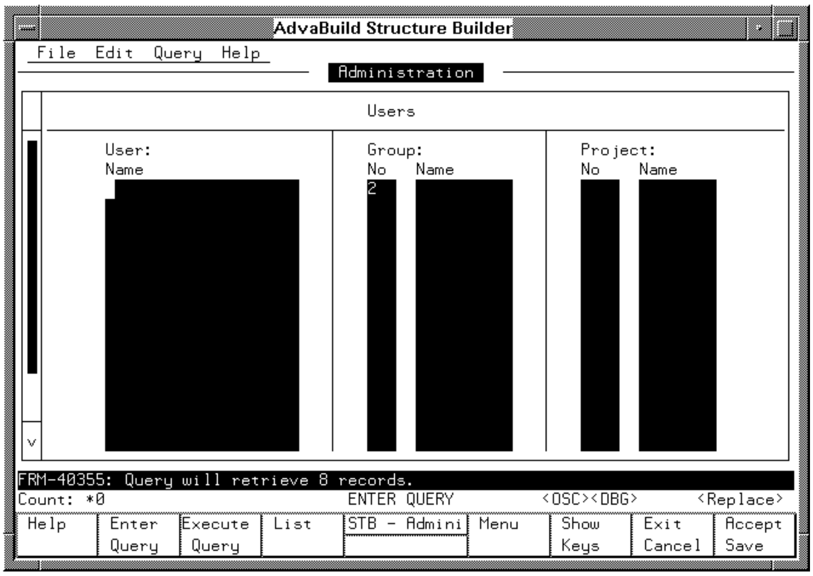
This dialog box has the following fields:

Parent object	Use this to query just the direct children of the selected object.
Whole subtree	Use this to query the whole subtree for the selected object.
Object Id	Use this field as a filter to retrieve any objects whose object ID matches a specific character string. When making an entry in a filter field, two characters can be used as wildcards. "%" sets the filter to match zero or more characters, and "_" sets the filter to match exactly one character. Examples: **CS1%** represents CS1 followed by any character string %.PID represents any character string followed by.PID CS_CM1 represents the character string CS_CM1 where the underscore can be any single character.
Object Type	Use this field as a filter to retrieve any objects whose object type matches the type specified in this field.

Click on the **Ok** button to create a list window with the specified list objects.

Query - Count Hits

Count Hits displays the number of items that satisfy the query, Figure 3-160.



```
FRM-40355: Query will retrieve 8 records.
Count: *0                              ENTER QUERY        <OSC><DBG>      <Replace>
```

Count Hits →

Figure 3-160. Count Hits

Query - Execute

Execute retrieves the items that satisfy the query (hits) and displays them. When you execute a query on a form that only displays one object (item) at a time, use the up and down arrow keys to scroll through the list of items that were retrieved.

Query - Cancel

If the query is cancelled, the original window is restored.

3.5.1.2 Help Menu

The **Help** menu is shown in Figure 3-161.

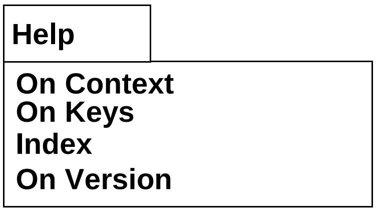


Figure 3-161. Help Menu

Help - On Context

"Sorry, Help on Context not yet implemented" is displayed.

Help - On Keys

Help On Keys shows the active keys for the window. For example:

Esc Esc m: Press <**ESC**> twice then press <**m**>

^**g**: Press <**CTRL**> then press <**g**> while holding down <**CTRL**>

InsertChar: Press <InsertChar>

Help - Index

Index displays an index of all available help topics. Navigate to the desired topic with the cursor keys and then select it by pressing <**Return**>.

Help - On Version

On Version shows the Version information of the software that is running.

3.5.2 Menus for the Administration Window

The Administration window provides access to the Projects and Users windows for creating and administering AdvaBuild projects and users.

3.5.2.1 Administration Window - File Menu

The **File** menu is shown in Figure 3-162.

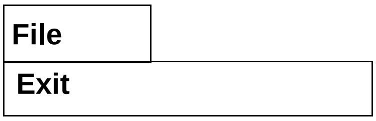


Figure 3-162. Administration Window - File Menu

File - Exit

Exit closes the Administration window.

3.5.2.2 Administration Window - Administer Menu

The **Administer** menu is shown in Figure 3-163.

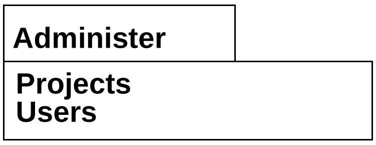


Figure 3-163. Administration Window - Administer Menu

Administer - Projects

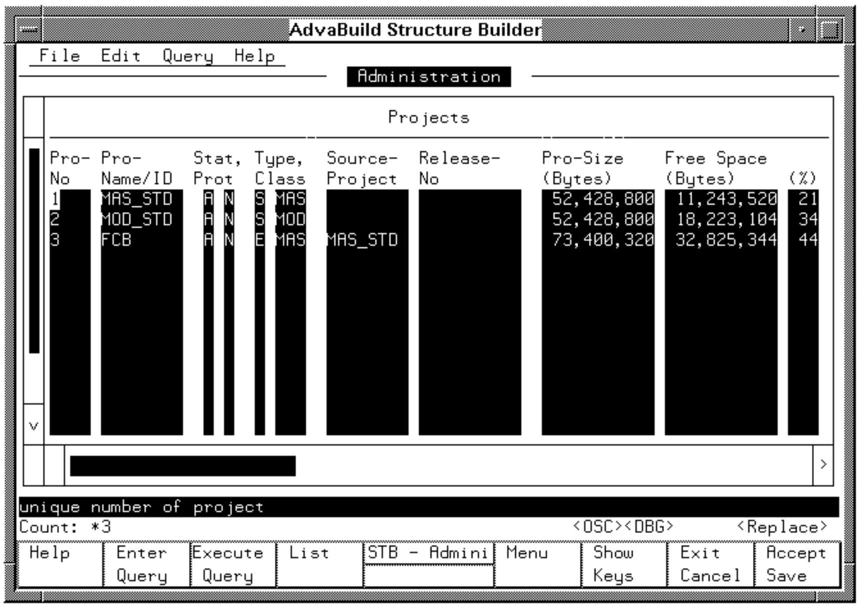
Projects opens a new window for displaying and manipulating the project-specific data, Figure 3-164.

Administer - Users

Users opens a new window for displaying and manipulating the user-specific data for all projects, Figure 3-174.

3.5.3 Menus for Projects Window

The Projects window provides a list of existing projects with project number and name, project status, protection, type and class, the source the project was copied from, information regarding the project size etc., Figure 3-164. The menu bar provides functions for inserting new projects, deleting old projects, changing the allotted size for existing projects, saving and restoring project releases and modifying the comment for the project node. This window is based on the FORMS user interface.



Pro-No	Pro-Name/ID	Stat, Prot		Type, Class		Source-Project	Release-No	Pro-Size (Bytes)	Free Space (Bytes)	(%)
1	MAS_STD	A	N	S	MAS			52,428,800	11,243,520	21
2	MOD_STD	A	N	S	MOD			52,428,800	18,223,104	34
3	FCB	A	N	E	MAS	MAS_STD		73,400,320	32,825,344	44

```
unique number of project
Count: *3                                   <OSC><DBG>      <Replace>
```

Figure 3-164. Projects Window

The fields of the Projects window are described below.

Pro-No	Project number.
Pro-Name/ID	Project name or project identification (unique).
Stat	Status of the project. This field reflects any administrative access to the projects and can have the following values: A for Accessible G for Generated C for Copied D for Deleted S for Saved R for Restored E for Enlarged
Prot	Project protection. If set to 'Y' = Yes, the project is protected against read access from other projects. As a result, it is not possible to copy parts of this project into another project.
Type	Project type, can have the values 'S' for standard project or 'E' for engineering project. A standard project is usually included in product delivery and cannot be deleted.
Class	Project class, can be either 'MAS' for Master projects or 'MOD' for MOD projects.
Source-Project	The source project the project was copied from (applies to engineering projects only as standard projects are created from scratch and not as a copy of another project.
Release No	If a project was restored from a previously saved release via the menu item **File Restore,** the number of the restored release is given in this field.
Pro-Size (Bytes)	The project size allocated for database files in Bytes.
Free Space (Bytes)	This shows the unused part of the allocated project size in Bytes.
Free Space (%)	Free space expressed as percentage of the total project size.
Start-Date	This field can be seen when moving from column to column via Tab and shows the date the project was created.
Project Description Line 1 ... Line 3	Text for a detailed description of the project, to be filled in during project insertion (see menu item Edit Insert), can be modified via TEB on project object.
Comment	In this field a short project description can be entered during project insertion. It can be modified by choosing menu item File Modify in the Projects window. This comment is shown in the box representing the project object of the STB tree presentation.

3.5.3.1 Projects Window - File Menu

The **File** menu is shown in Figure 3-165.

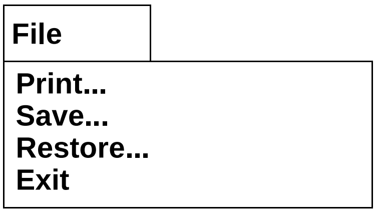
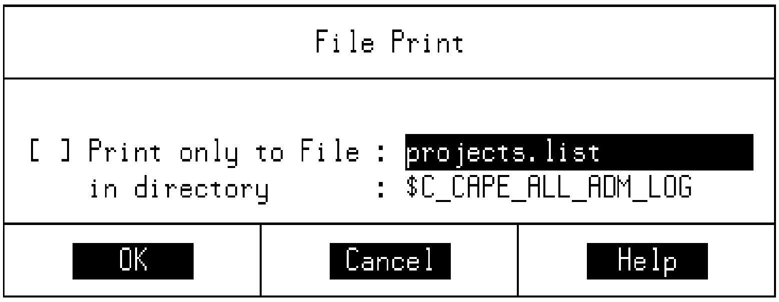


Figure 3-165. Projects Window - File Menu

File - Print

Print prints out project information in a file and, optionally, on the printer. **Print** displays the Print Project dialog box, Figure 3-166. To print project details only to a file and not to the printer as well, enter 'X' in the first input field. Fill in the file name. If no extension is given, the extension '.list' will be appended automatically. When the cursor is positioned on **OK**, select **Accept/Save** to commit your entries.


File Print

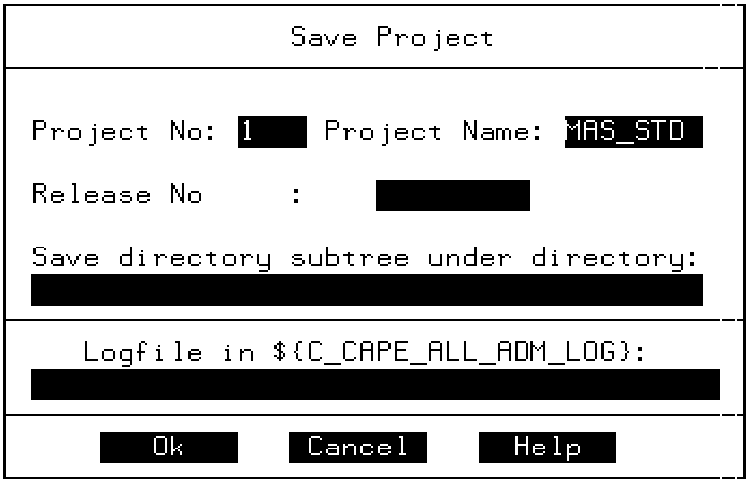
[ ] Print only to File : projects.list
in directory : $C_CAPE_ALL_ADM_LOG

Figure 3-166. Print Projects Dialog Box

File - Save

Save can be used to 'freeze' a copy of a certain project.

Save displays the Save Project dialog box, Figure 3-167. To save a project, define the save parameters by entering the required data in the dialog box as described below. When the cursor is positioned on **OK**, select **Accept/Save** to commit your entries and start the program to save the project.



Project No: 1 Project Name: MAS_STD

Release No :

Save directory subtree under directory:

Logfile in ${C_CAPE_ALL_ADM_LOG}:

Figure 3-167. Save Project Dialog Box

The fields of the Save dialog box are described below.

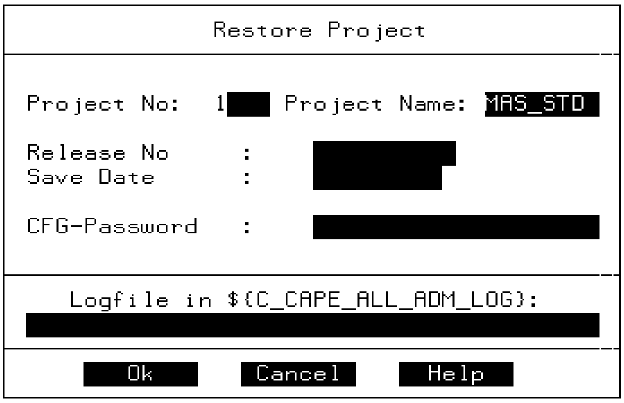
Project No	This field is used to specify the project to be saved. Enter the project number assigned to the project when it was created. You can use the **List** function (<**f4**>) to call up a list of valid project numbers to facilitate selection of a project number.
Project Name	The project is automatically entered when you select Project No.
Release No	This is a number or a character string that you assign to this version of the project when you export it to distinguish it from other versions.
Save Directory subtree under directory	Enter the path specification for the target directory. If you want to write to another disc make sure to give the complete disc- and path name e.g. /disc2/dir_1/dir_2. A default pathname on the project disc will be given as soon as you enter a release identifier. It can be accepted by simply proceeding to Ok.

File - Restore

File Restore is used to restore a project saved with the Save menu item. **Restore** displays the Restore Project dialog box, Figure 3-168. To restore a project release, define the restore parameters by entering the required data in the dialog box as described below. When the cursor is positioned on **OK**, select **Accept/Save** to commit your entries and start project restoration.

NOTE

All modifications done to the project since the release to be restored was drawn will be lost after restoring this release. If you want to keep these modifications, save a new release first before restoring an old one.



Project No: 1 Project Name: MAS_STD

Release No :

Save Date :

CFG-Password :

Logfile in ${C_CAPE_ALL_ADM_LOG}:

Figure 3-168. Restore Project Dialog Box

Project No:	This field is used to specify the project to be restored. Enter the project number assigned to the project when it was created. You can use the **List** function (<**f4**>) to select a project number. A released version of a project can only be restored to the source project it was copied from.
Project Name	The project is automatically entered when you select Project No.
Release No:	Enter the release identifier of the project being restored (release identifier was assigned when project was saved and is selectable via **List** function (<**f4**>)).
Save Date	The save date is entered automatically when you select Release No.
CFG-Password	Enter the password for the project CFG user.

File - Exit

Exit closes the Projects window.

3.5.3.2 Projects Window - Edit Menu

The **Edit** menu is shown in Figure 3-169.

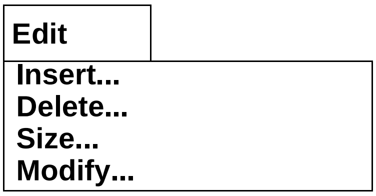
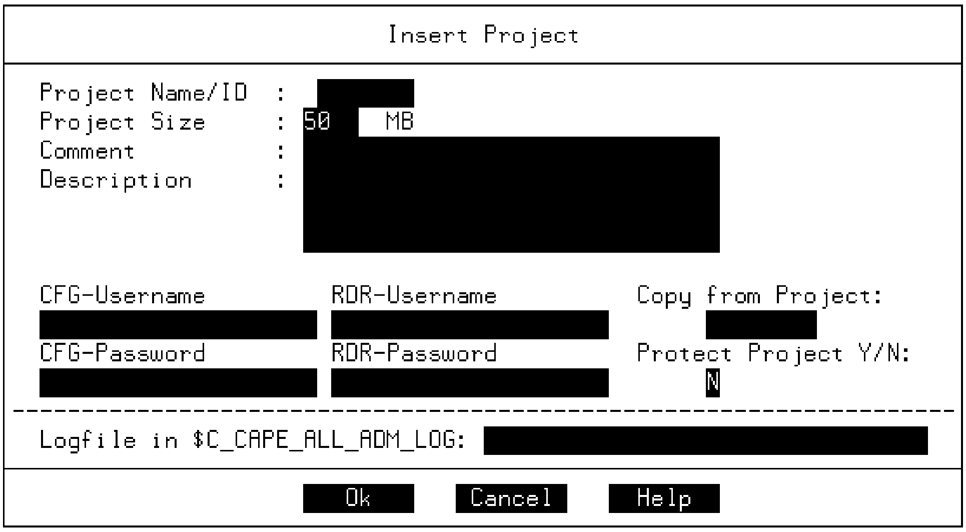


Figure 3-169. Projects Window - Edit Menu

Edit - Insert

Insert creates a new project by copying a source project. **Insert** displays the Insert Project dialog box, Figure 3-170. To insert a new project, define the project parameters by entering the required data in the dialog box as described below. When the cursor is positioned on **OK**, select **Accept/Save** to commit your entries and start the project insertion program.



Insert Project

Project Name/ID :
Project Size : 50 MB
Comment :
Description :

CFG-Username
CFG-Password
RDR-Username
RDR-Password
Copy from Project:
Protect Project Y/N: N

Logfile in $C_CAPE_ALL_ADM_LOG:

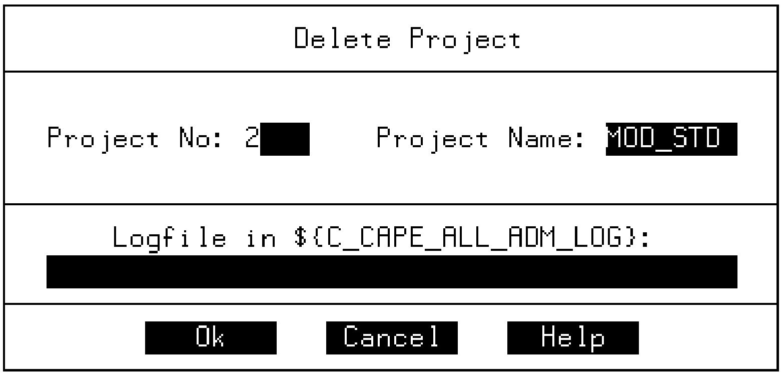
Figure 3-170. Insert Project

The fields in the Insert project dialog box are described below.

Project Name/ID	Enter a unique project name in this field. The name you enter here is used to identify the project in other dialog boxes (see **Save** and **Restore** menu items for examples). Note that once you have assigned a name to a project this name cannot be changed anymore.
Project Size	Enter the amount of disc space required for the database file or simply accept the default value of 50 MByte. In any case will the size be adapted to the used up space of the source project after the source's name has been filled in (see below, Copy from Project). Example: if the source project has the size 80 MB but is filled with only 60 MB, the size of the new project will be set to slightly over 60 MB. Note that if the database file space is filled up with data elements (see columns *Size* in the Projects window), an additional database file has to be created using the **Size** menu item in the **Edit** menu.
Comment	This field is optional. You can use this field to enter a comment for a short project description to be seen in the STB's tree presentation of the project object. To modify the comment choose the **Modify** menu item in the **Edit** menu.
Description	This field is optional and consists of 3 lines. You can use this field to enter a text for a more detailed description of the project.
CFG and RDR Username &Password	Two users are automatically assigned to the project: a CONFIG user (name '<Project_Name>_CFG') and a READER user (name '<Project_Name>_RDR'). These names are predefined and cannot be modified whereas the passwords can changed if so desired. The READER user will have only read access to the project.
Copy from Project	Enter the project being used as the source for the new project.
Protect Project Y/N	If no read access for users of other projects should exist, choose Protect Project = Y. In this case it will not be possible for users of other projects to copy parts of the new project into their own project via STB **File Import.** The default value is N = No so users of other projects can copy project parts to their projects. If you want to toggle the project protection from Yes to No or vice versa, choose menu item **Modify** from the **Edit** menu.

Edit - Delete

Delete is used to delete the whole project-directory-subtree including the database files, and to remove the project data from the database. **Delete** displays a dialog box for specifying the project to be deleted, Figure 3-171. When you enter the project number, the project name and log file are automatically filled in. When the cursor is positioned on **OK**, select **Accept/Save** to commit your entries and start the project deletion program.



Delete Project

Project No: 2

Project Name: MOD_STD

Logfile in ${C_CAPE_ALL_ADM_LOG}:

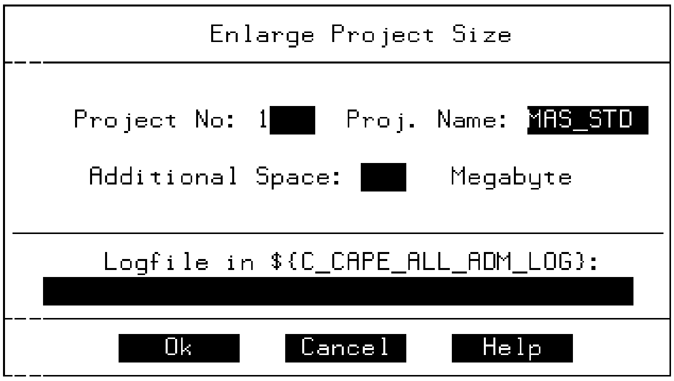
Figure 3-171. Delete Project Dialog

NOTE

- Standard projects (marked with 'S' in column *Type* of the Projects window) cannot be deleted.
- Do not delete the project subtree with operating system commands. This will cause the database to crash.

Edit - Size

Size is used to create an additional database file for a project whose file has become too small or has been filled. To find out a project's total size and used space, see the Projects window.**Size** displays a dialog box for entering the project name (can be chosen from list) and the additional space for the database, Figure 3-172. When you enter the project number, the project name and log file are automatically filled in. When the cursor is positioned on **OK**, select **Accept/Save** to commit your entries and start the program to enlarge the project.



Enlarge Project Size

Project No: 1

Proj. Name: MAS_STD

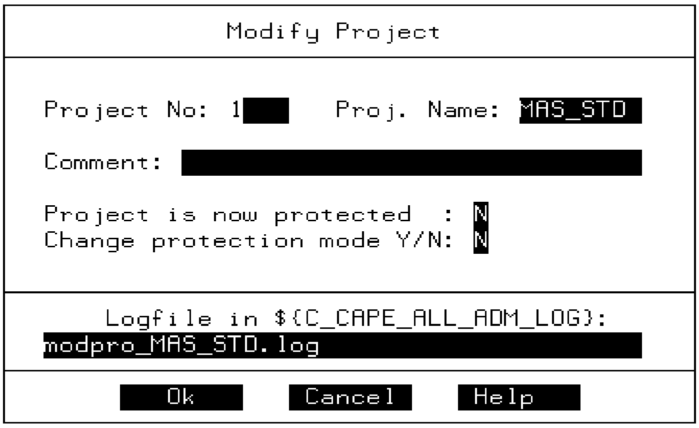
Additional Space: Megabyte

Logfile in ${C_CAPE_ALL_ADM_LOG}:

Figure 3-172. Enlarge Project Size Dialog Box

Edit - Modify

Modify is used to modify the comment on the project object which will be displayed in the Structure Builder's tree representation and the protection of the project. **Modify** displays a dialog box for entering the project name (can be chosen from a list), the new comment, and, optionally, 'Y' or 'N' for toggling the project's present protection mode which is shown in field 'Project is now protected Y/N', Figure 3-173. When you enter the project number, the project name and log file are automatically filled in. When the cursor is positioned on **OK**, select **Accept/Save** to commit your entries and start the project modification program.



Project No: 1 Proj. Name: MAS_STD

Comment:

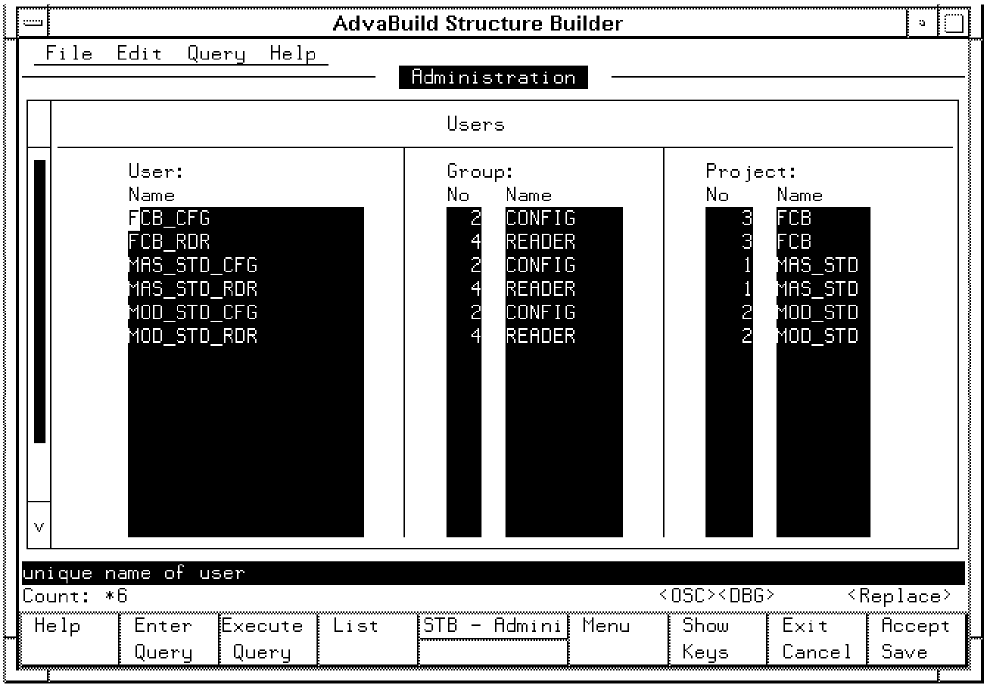
Project is now protected : N
Change protection mode Y/N: N

Logfile in ${C_CAPE_ALL_ADM_LOG}:
modpro_MAS_STD.log

Figure 3-173. Modify Project Size Dialog Box

3.5.4 Menus for Users Window

The Users window provides a list of users with user name, group number and name, and project number and name, Figure 3-174. The entries cannot be changed directly in this window.



User: Name	Group: No	Group: Name	Project: No	Project: Name
FCB_CFG	2	CONFIG	3	FCB
FCB_RDR	4	READER	3	FCB
MAS_STD_CFG	2	CONFIG	1	MAS_STD
MAS_STD_RDR	4	READER	1	MAS_STD
MOD_STD_CFG	2	CONFIG	2	MOD_STD
MOD_STD_RDR	4	READER	2	MOD_STD

unique name of user
Count: *6 <OSC><DBG> <Replace>

Figure 3-174. Users Window

3.5.4.1 Users Window - File Menu

The **File** menu is shown in Figure 3-175.

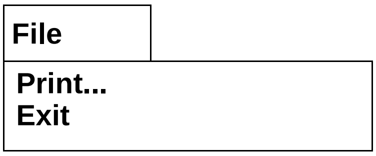
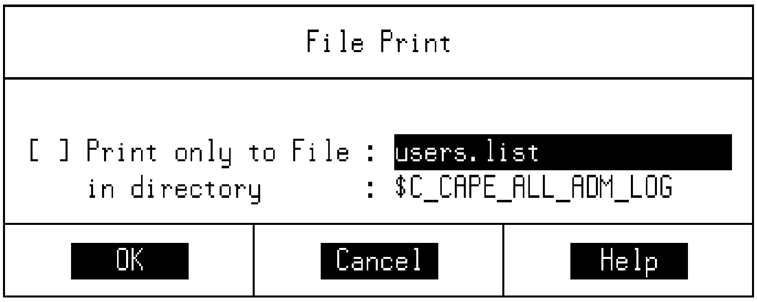


Figure 3-175. Users Window - File Menu

File - Print

Print prints out all data of Users window in a file and, optionally, on the printer. **Print** displays the Print Users dialog box, Figure 3-176. To print user details only to a file and not to the printer as well, enter 'X' in the first input field. Fill in the file name. If no extension is given, the extension '.list' will be appended automatically. When the cursor is positioned on **OK**, select **Accept/Save** to commit your entries and start the print program.



[ ] Print only to File : users.list
in directory : $C_CAPE_ALL_ADM_LOG

Figure 3-176. Print Users Dialog Box

File - Exit

Exit closes the Users window.

3.5.4.2 Users Window - Edit Menu

The **Edit** menu is shown in Figure 3-177.

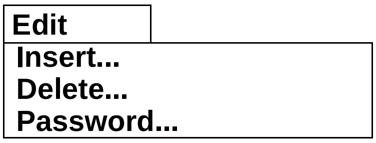


Figure 3-177. Users Window - Edit Menu

Edit - Insert

Insert is used to establish a new user. **Insert** displays a dialog box for entering the parameters, Figure 3-178. To insert a new user, define the user parameters by entering the required data in the dialog box as described below. When the cursor is positioned on **OK**, select **Accept/Save** to commit your entries and start the user insertion program.

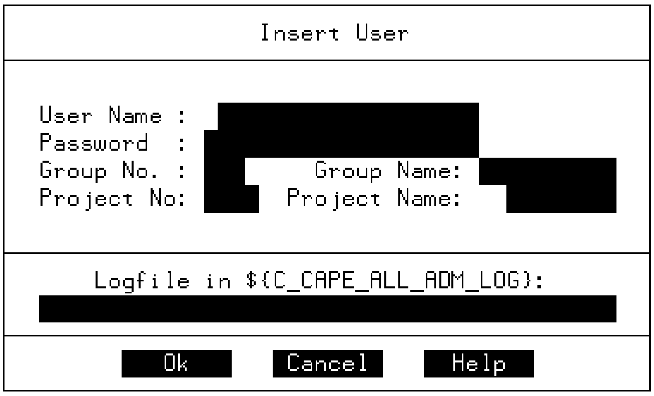


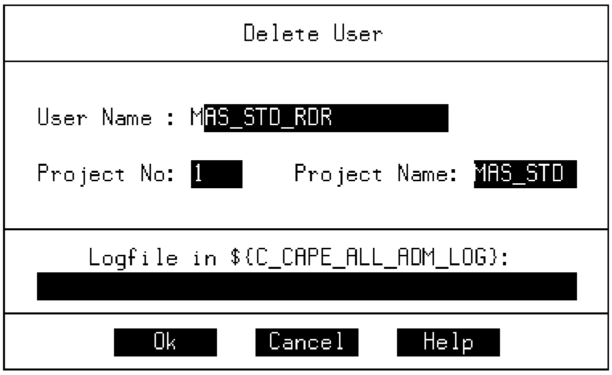
Figure 3-178. Insert User Dialog Box

The fields in the Insert Users dialog box are described below.

User Name	Use this field to enter a user name. Note that the user name should be unique in the system. It should start with a letter and continue with letters, digits or '_' but not contain any special characters.
Password	Use this field to enter a user password. As to the structure of the password, the same restrictions apply as to user names.
Group No.	Group No. determines the user access rights as indicated below: **1** = ADMIN (overall administrator, there should be only one ADMIN user) **2** = CONFIG (Administrator, normally one per project) **3** = WRITER (User that works with the project data) **4** = READER (User that has read-only access to the project data) Group number is selectable from a list (via **List** function <**f4**>).
Project No.	The project the user belongs to is identified by the Project Number. This item can be chosen via the **List** function. Note that a user can belong to only one project.

Edit - Delete

Delete is used to delete a user from a project and revoke the user's access to the system.**Delete** displays a dialog box for entering the name of the user to be deleted, Figure 3-179. After filling in the required fields press Accept/Save to commit your entries and start the user deletion program.



Delete User

User Name : MAS_STD_RDR

Project No: 1 Project Name: MAS_STD

Logfile in ${C_CAPE_ALL_ADM_LOG}:

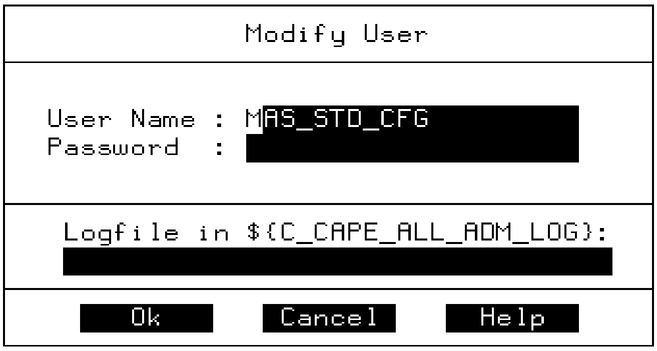
Figure 3-179. Delete User Dialog Box

This dialog box has the following fields:

User Name	Use this field to specify the name of the user to be deleted.
Project No.	When you enter the user name, the number of the project that the user is associated with is automatically entered.
Project Name	When you enter the user name, the name of the project that the user is associated with is automatically entered.
Log File	This field is filled in automatically.

Edit - Password

Passwords are assigned when the user is created. **Password** is used to change a user password. **Password** displays the Modify User dialog box, Figure 3-180. This dialog box asks for the user's name and new password (username can be selected from list).



Modify User

User Name : MAS_STD_CFG
Password :

Logfile in ${C_CAPE_ALL_ADM_LOG}:

Figure 3-180. Modify User Dialog Box

3.5.5 Menus for Structure Builder Tree and List Presentation Windows

The menu descriptions in this section are applicable for menu items in the Tree and List Presentation Windows, and the Object Information Window. Menu items that are common to these windows operate in the same manner.

3.5.5.1 File Menu

The **File** menu is shown in Figure 3-181.

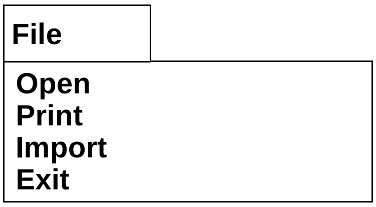


Figure 3-181. Structure Builder - File Menu

File - Open

The **Open** item in the **File** menu is used to log in to a specific project, or to start the administration tool. **Open** displays a dialog box for entering your username and password, Figure 3-182.

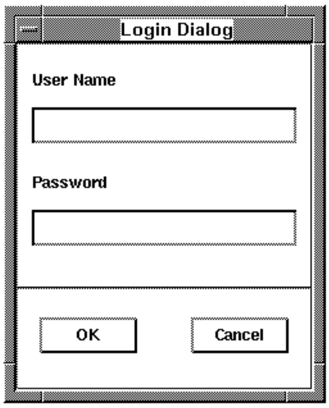


Figure 3-182. Project Log In Dialog Box

This dialog box has the following fields:

User name	Use this field to enter either the Administrator's user name, or a project-specific user name.
Password	Use this field to enter either the Administrator's password, or a project-specific password.

To start the administration tool enter a the Administrator user name and password, and then click on **Login**.

To open a project, enter your project user name and password, and then click on **Login**. This calls up the root object and its next level (direct children) in the structure of your project.

File - Print

Print is used to print information such as object identifier, object type, and comment for objects in the Structure Builder buffer. To print objects, first select an object. Then choose **Print** from the **File** menu. This displays a dialog box, Figure 3-183.

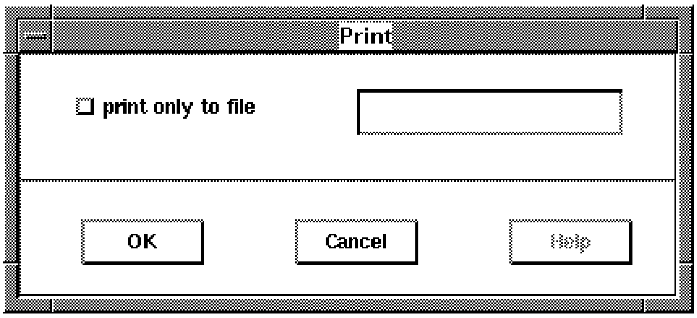


Figure 3-183. Print Dialog Box

This dialog box has the following check button:

print only to file	Use this field to create only a file in the AdvaBuild user directory where the print file will be stored.

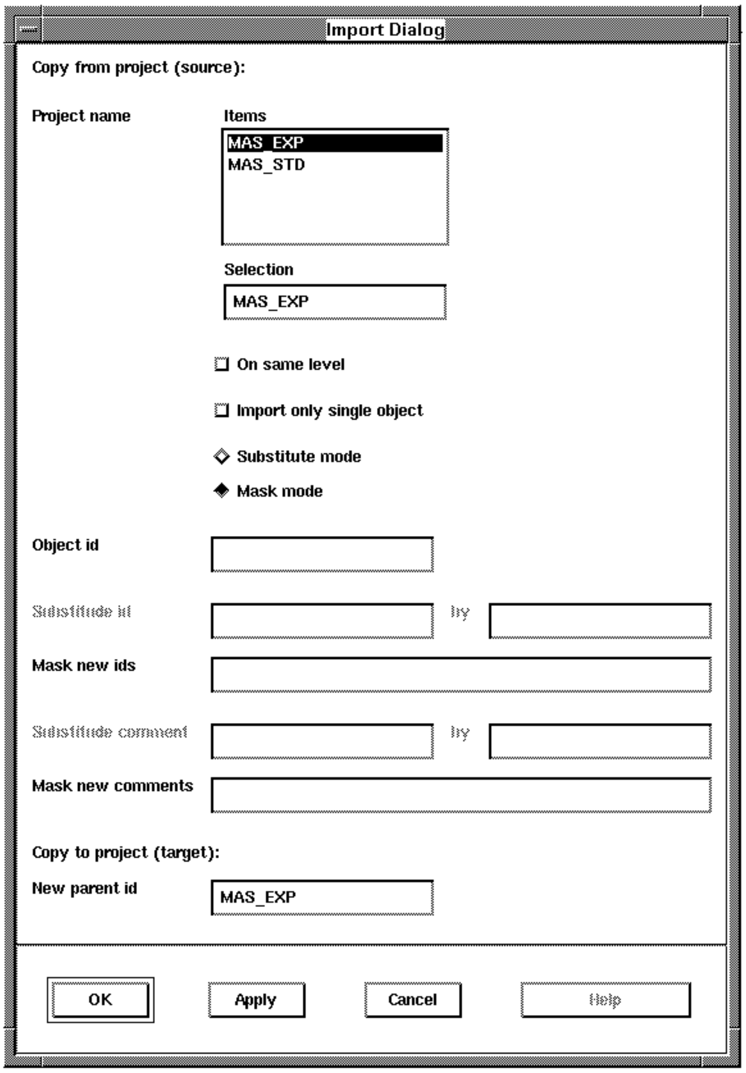
You can use this dialog box to either print the selected object, or save it to a file.

To print the object, press the **OK** button without any changes in the dialog box. This creates a temporary post script file that is sent directly to the standard postscript printer queue of the Advant Station.

To create a text file for the object, click on the **print only to file** button, enter a file name and press the **OK** button. This creates a text file in the AdvaBuild user directory of the Structure Builder. This file is not sent directly to a printer queue. The contents of this file can be shown by starting the HP VUE file manager (choose **File Manager** from the **Utility** menu**,** and then choose **View** from the **Actions** menu). The file contains the displayed object structure in the tree window with object identifier, object type and comment. The structure level is shown by indentation.

File - Import

Import is used to copy a subtree from another project into the current project. When you select **Import**, the following dialog box is displayed, Figure 3-184.



Copy from project (source):

Project name

Items:
MAS_EXP
MAS_STD

Selection: MAS_EXP

☐ On same level

☐ Import only single object

○ Substitute mode

● Mask mode

Object id:

Substitute id: by

Mask new ids:

Substitute comment: by

Mask new comments:

Copy to project (target):

New parent id: MAS_EXP

Figure 3-184. Import Project

This dialog box has the following fields:

Copy from project (source)	This label indicates the fields below are for the source project.
Project name	This selection box displays a list of project names from which you can choose the source project. Click on a name to select.
Selection	This field shows the selected project.
On same level	By default imported objects are inserted at the bottom of the selected subtree. On same level lets you specify where to insert objects in the subtree. This is particularly useful when you need to specify the execution order of the object being imported. If execution order is not critical, then do not use this check box.
Copy only single object	Use this button when you want to import just the specified object without its subtree. If this button is not selected (default condition), the object and its whole subtree are imported.
Object id	Use this field to enter the object id of the selected project that you are importing.
Substitute mode	
Mask mode	These are two mutually exclusive modes for masking object IDs and comments of duplicate objects. You must choose one or the other. The default condition is the mask mode. The Mask Mode replaces the characters that occupy a specific position in the object ID with a specified mask (for example, replace the ninth and tenth characters with **25**). The Substitute Mode replaces a character string that you specify, with a new character string (for example replace **B100** with **C125**). This is the easiest mode to use; however, you need to be aware that every instance of the specified character string will be replaced. If it occurs more than once in the object ID or comment, and you only want it replaced in one spot, then you must use the mask mode.
Copy to project (target)	This label indicates the field below is related to the target project.
New parent id	Use this field to enter the object id for the parent object where you want to insert the subtree that you are importing.

Click on **Apply** to import the object and then return to the dialog box to continue importing objects, or click on **OK** when you are finished. Importing a large subtree may be time-consuming.

File - Exit

Exit ends the Structure Builder session and closes the tree and all list windows. To exit, choose **Exit** from the **File** menu. All actions executed during the current session are always saved automatically, so there is no need to save prior to exiting.

3.5.5.2 Edit Menu

The **Edit** menu is shown in Figure 3-185.

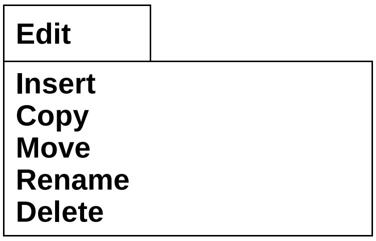
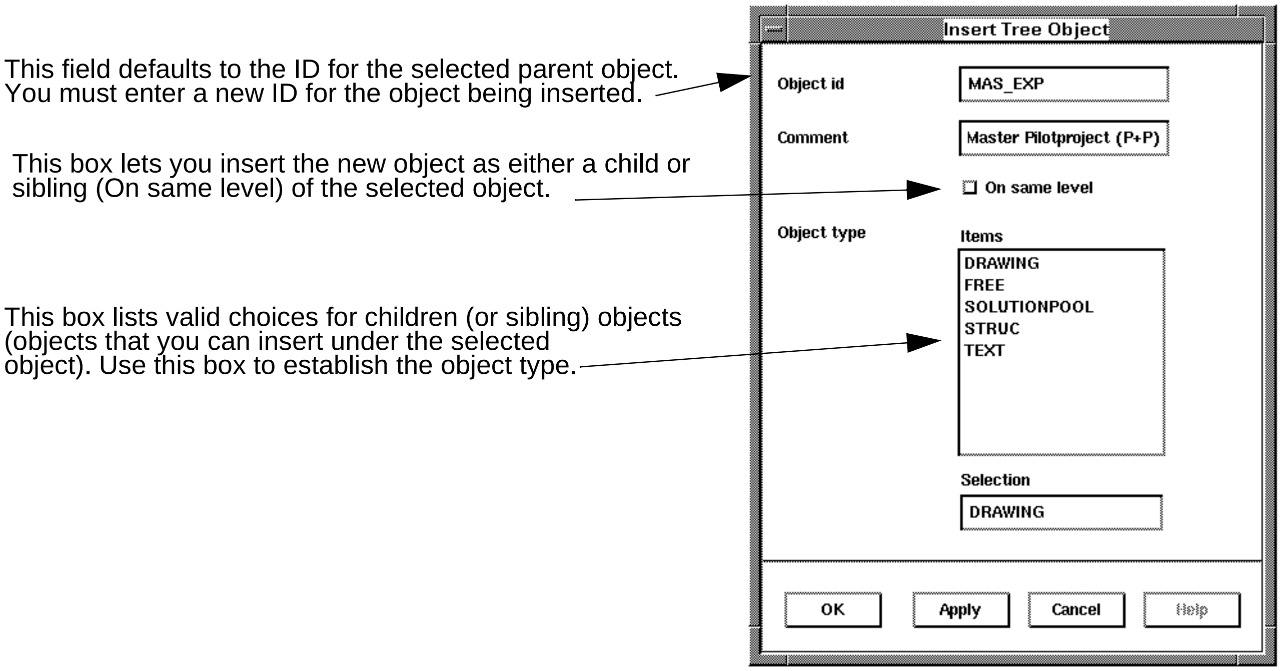


Figure 3-185. Structure Builder - Edit Menu

Edit - Insert

To insert an object, select the relative (parent or sibling) object for the new object, then choose **Insert** from the **Edit** menu. This displays the following dialog box, Figure 3-186.



Object id: MAS_EXP
Comment: Master Pilotproject (P+P)
☐ On same level
DRAWING
FREE
SOLUTIONPOOL
STRUC
TEXT
Selection: DRAWING

This field defaults to the ID for the selected parent object. You must enter a new ID for the object being inserted.

This box lets you insert the new object as either a child or sibling (On same level) of the selected object.

This box lists valid choices for children (or sibling) objects (objects that you can insert under the selected object). Use this box to establish the object type.

Figure 3-186. Insert Object Dialog Box

This dialog box has the following fields:

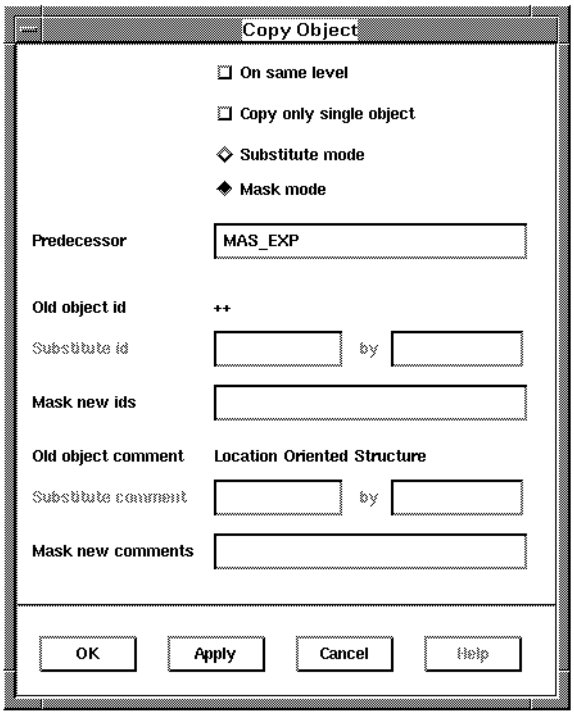
Object id	This field displays the ID of the selected object. Use this field to enter a unique object ID for the object that you are inserting.
Comment	This field is optional. Use this field to enter a comment that describes the new object. The comment is displayed in the rectangular box that represents the object in the tree presentation.

On same level	Use this check box to insert a a sibling of the selected object (on the same level). The default (on same level not selected) sets up the dialog for inserting a child of the selected object. If you want to insert a sibling, click on this check box.
Object Type	The contents of this box is governed by the On same level button. When you select On same level, this box lists object types that you can insert as siblings of the selected object. When you do not select On same level, this box lists object types that you can insert as children of the selected object. If the list is larger than the box, a scroll bar is provided. Click on an object type to select.
Selection	This field shows the currently selected object type.

Click on **Apply** to insert the object and then return to the dialog box to continue inserting objects, or click on **OK** to insert the object and then close the dialog. Object attributes are set at their default values. These can be edited via the Template Builder.

Edit - Copy

Copy is used to copy a selected object and its subtree and insert the duplicate copy under a specified parent object within the current project. First select the object that want to copy, and then choose **Copy** from the **Edit** menu. **Copy** displays the following dialog box, Figure 3-187.



Copy Object

☐ On same level
☐ Copy only single object
○ Substitute mode
● Mask mode

Predecessor: MAS_EXP
Old object id: ++
Substitute id ______ by ______
Mask new ids: ______
Old object comment: Location Oriented Structure
Substitute comment ______ by ______
Mask new comments: ______

Figure 3-187. Copy Object

This dialog box has the following fields:

On same level	By default copied objects are inserted at the bottom of the selected subtree. On same level lets you specify where to insert objects in the subtree. This is particularly useful when you need to specify the execution order of the object being copied. If execution order is not critical, then do not use this check box.
Copy only single object	Use this button when you want to copy just the selected object without its subtree. If this button is not selected (default condition), the object and its whole subtree are copied.
Substitute mode	
Mask mode	These are two mutually exclusive modes for masking object IDs and comments of duplicate objects. You must choose one or the other. The default condition is the mask mode. The Mask Mode replaces the characters that occupy a specific position in the object ID with a specified mask (for example, replace the ninth and tenth characters with **25**). The Substitute Mode replaces a character string that you specify, with a new character string (for example replace **B100** with **C125**). This is the easiest mode to use; however, you need to be aware that every instance of the specified character string will be replaced. If it occurs more than once in the object ID or comment, and you only want it replaced in one spot, then you must use the mask mode.
Predecessor	Initially, this field defaults to the object ID for the parent of the selected object. If you selected On same level, you can use this field to specify where in the execution order you want to insert the duplicate object and subtree. To do this, enter the ID of the object that you want to insert the duplicate under (see example under On same level). If you did not select On same level, you can use this field to specify a different parent to insert the duplicate object under, or you can use the default. In either case, the duplicate object and subtree are added to the bottom of the specified subtree.
Old object id	This field displays the ID of the selected object that you are copying. It is for reference to facilitate definition of a mask.
Substitute id	When you select Substitute Mode, specify the character string to be replaced in the first field and the new (substitute) character string in the second field. Every instance of the specified character string will be replaced. For instance, if the original object IDs all have B100 for a suffix, and you want the duplicate objects to have a suffix of **C125**, enter **B100** by **C125**.

Mask new ids	When you select Mask Mode, use this field to define the mask. The mask is applied by character position in the object ID rather than by specific characters. As a prerequisite, the descendents should all have object IDs that closely resemble the parent object ID. Generally, they share a common prefix (for example, CS1, CS1_CM1, CS1_CM11_I/O, CS1_CM1_CCF, and so on). This way, you can define a mask by simply changing one character. In the example above, if you enter CS2 (or press the spacebar twice and then enter **2**), the duplicate objects will have the following IDs (CS2, CS2_CM1, CS2_CM1_I/O, CS2_CM1_CCF
Old object comment	This field displays the comment of the selected object that you are copying. It is for reference to facilitate definition of a mask.
Substitute comment	When you select Substitute Mode, specify the character string to be replaced in the first field and the new (substitute) character string in the second field. Every instance of the specified character string will be replaced.
Mask new comments	When you select Mask Mode, use this field to define the mask. The mask is applied by character position in the comment rather than by specific characters.

When you finish defining the copy parameters, click on either **OK** or **Apply**. Clicking on **OK** applies the copy parameters and then removes the dialog box from the screen. **Apply** applies the copy parameters and leaves the dialog box so you can continue to copy other objects. Any of the current values can be used for the next action. When you copy a subtree with links, links that are open in the original subtree are deleted in the duplicate, and links that are closed in the original subtree are open in the duplicate.

Edit - Move

Move is used to disconnect a subtree from one parent and connect it to another. You can also move an object up or down in the execution order of its current subtree. First select the highest level object in the subtree to be moved, and then choose **Move** from the **Edit** menu. **Move** displays the following dialog box, Figure 3-188. Click on **OK** when you are finished with this dialog.

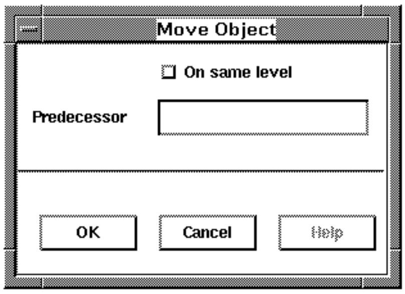


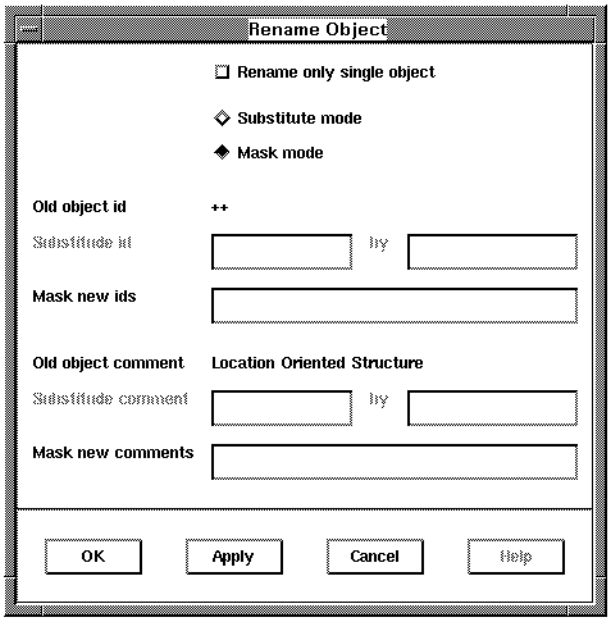
Figure 3-188. Move Object

This dialog box has the following fields:

On same level	Click on this check box to move the selected object up or down in the execution order of its current subtree. If you want to move the selected object to a new parent, then do not use this check box.
Predecessor	This field defaults to the object ID for the parent of the selected object. If you selected On same level, you can use this field to specify where to put the selected object in the execution order. To do this, enter the ID of the object that you want to move the selected object under (see example under On same level). If you did not select On same level, you can use this field to specify a different parent to move the selected object under.

Edit - Rename

Rename is used to change the object ID and comment. **Rename** operates on a single object or on a whole subtree. First select the object to be renamed, and then choose **Rename** from the **Edit** menu. **Rename** displays a dialog box for entering the rename parameters, Figure 3-189.



☐ Rename only single object

◇ Substitute mode

◆ Mask mode

Old object id ++

Old object comment Location Oriented Structure

Figure 3-189. Rename Object

This dialog box has the following fields:

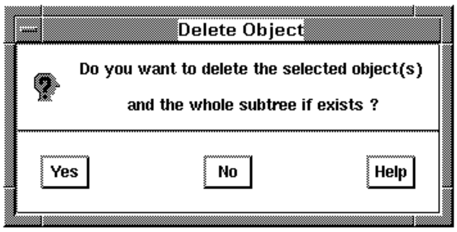
Rename only single object	Use this button when you want to rename just the selected object without its subtree. If this button is not selected (default condition), the object and its whole subtree are renamed.
Substitute mode	
Mask mode	These are two mutually exclusive modes for masking object IDs and comments of duplicate objects. You must choose one or the other. The default condition is the mask mode. The Mask Mode replaces the characters that occupy a specific position in the object ID with a specified mask (for example, replace the ninth and tenth characters with **25**). The Substitute Mode replaces a character string that you specify, with a new character string (for example replace **B100** with **C125**). This is the easiest mode to use; however, you need to be aware that every instance of the specified character string will be replaced. If it occurs more than once in the object ID or comment, and you only want it replaced in one spot, then you must use the mask mode.
Old object id	This field displays the object ID of the selected object that you are renaming. This is provided as reference to facilitate definition of a mask.
Substitute id	When you select Substitute Mode, specify the character string to be replaced in the first field and the new (substitute) character string in the second field. Every instance of the specified character string will be replaced. For instance, if the original object IDs all have B100 for a suffix, and you want the duplicate objects to have a suffix of **C125**, enter **B100** by **C125**.
Mask new ids	When you select Mask Mode, use this field to define the mask. The mask is applied by character position in the object ID rather than by specific characters. As a prerequisite, the descendents should all have object IDs that closely resemble the parent object ID. Generally, they share a common prefix (for example, CS1, CS1_CM1, CS1_CM11_I/O, CS1_CM1_CCF, and so on). This way, you can define a mask by simply changing one character. In the example above, if you enter CS2 (or press the spacebar twice and then enter **2**), the duplicate objects will have the following IDs (CS2, CS2_CM1, CS2_CM1_I/O, CS2_CM1_CCF
Old object comment	This field displays the comment of the selected object that you are renaming. This is provided as reference to facilitate definition of a mask.

Substitute comment	When you select Substitute Mode, specify the character string to be replaced in the first field and the new (substitute) character string in the second field. See description of Substitute ID above.
Mask new comments	When you select Mask Mode, use this field to define the mask. The mask is applied by character position in the comment rather than by specific characters.

When you are finished defining the rename parameters, click on either **OK** or **Apply**. Clicking on **OK** applies the rename parameters and then removes the dialog box from the screen. **Apply** applies the rename parameters and leaves the dialog box so you can continue to rename other objects. Any of the current values can be used for the next action.

Edit - Delete

Delete is used to delete an object and its entire subtree. First select the object that you want to delete, and then choose **Delete** from the **Edit** menu. **Delete** displays a dialog box for either confirming or canceling the delete function, Figure 3-190. Click on **Yes** to delete the selected object with all it's whole subtree. Open links are removed, and closed links are opened.



Do you want to delete the selected object(s)
and the whole subtree if exists ?

Figure 3-190. Delete Object Dialog Box

3.5.5.3 View Menu

The **View** menu is shown in Figure 3-191.

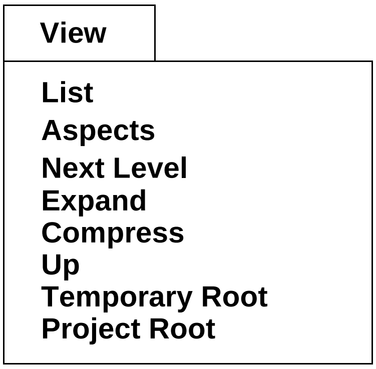
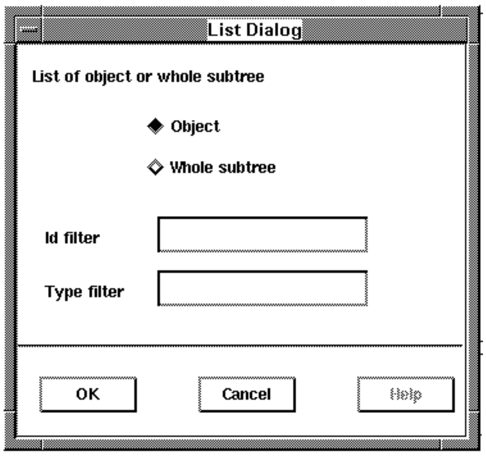


Figure 3-191. Structure Builder - View Menu

View - List

List is used to display the list objects of a selected tree object in a separate list window. List objects can only be created by the **Insert** function in the list window; however, you can remove tree objects from the tree presentation and place them in the list presentation. This functionality is supported by the Object Information window.

To display list objects, select the tree object whose list objects you want to view, then choose **List** from the **View** menu. This displays the following dialog box, Figure 3-192.



List Dialog

List of object or whole subtree

◆ Object

◇ Whole subtree

Id filter ☐

Type filter ☐

Figure 3-192. List of Objects

This dialog box has the following fields:

Object	Use this button to display list objects that are the direct children of the selected tree object. This is the default setting.
Whole subtree	Use this button to display the whole subtree (all descendants that are list objects) for the selected tree object.
Id filter	Use this field to define a specific subset of list objects to display, based on the object ID. When you make an entry in this field, the list window displays all list objects whose object ID matches the text string defined in this field.
Type filter	Use this field to define a specific subset of list objects to display, based on the object type. When you make an entry in this field, the list window displays all list objects whose object type matches the type specified in this field.

You must select either Object or Whole subtree. Filters are optional. You can use any combination of filter fields (one, both, or none) in conjunction with the Object or Whole Subtree. When making an entry in a filter field, two characters can be used as wildcards. "%" sets the filter to match zero or more characters, and "_" sets the filter to match one character.

Examples

CS1% represents CS1 followed by any character string

%.PID represents any character string followed by.PID

CS_CM1 represents the character string CS_CM1 where the underscore can be any character.

Click on the **OK** button to display a list window with the specified list objects.

View - Aspects

Aspects opens the Solution Aspect window. This only works if the selected object have a Solution state button.

View - Next Level

Next Level expands the tree presentation to show the next level of a selected tree object. Select an object that you want to show the next level for, and then choose **Next Level** from the **View** menu. This expands the view to show direct children of the selected object. This function is equivalent to clicking the selected object's subtree button.

View - Expand

Expand expands the tree presentation to show the whole subtree of a selected tree object (children, children's children, and so on). Select an object that you want to show the whole subtree for, and then choose **Expand** from the **View** menu. This function is equivalent to holding down <**shift**> and clicking on the selected object's subtree button.

View - Compress

Compress hides all objects under the selected object. Select an object whose subtree you want to hide, and then choose **Compress** from the **View** menu. This removes all objects in the subtree of the selected object. This function is equivalent to clicking the selected object's subtree button, when parts of the subtree are expanded.

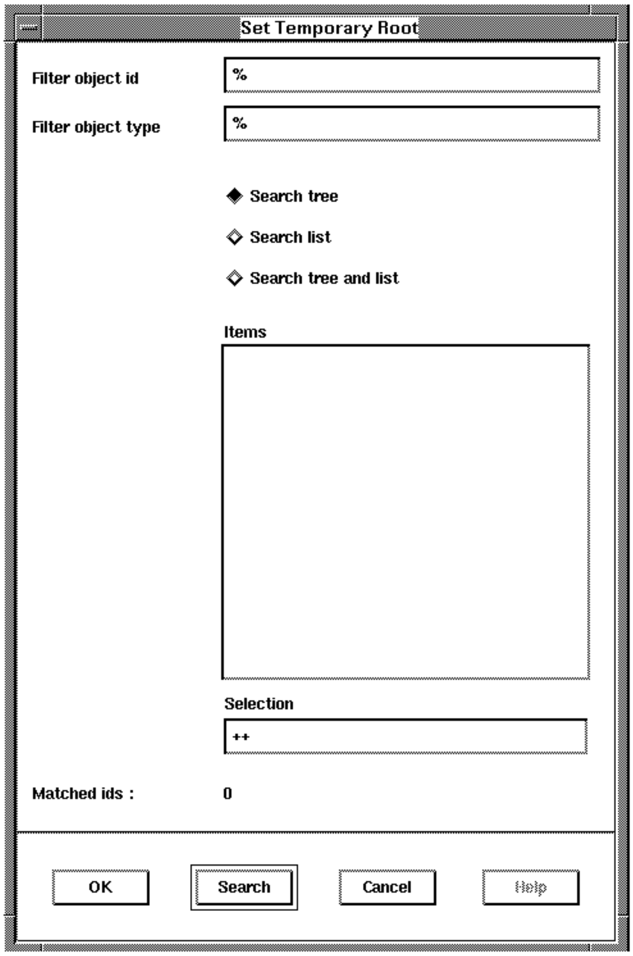
View - Up

Up sets the parent of the temporary root to the new temporary root. If a temporary root is set, choose **Up** from the **View** menu. This sets the new temporary root, and updates the display to show the temporary root and its next structure level (direct children). This function is equivalent to clicking the temporary root object's parent button. Thus you can navigate backwards step by step in direction of the project root object.

View - Temporary Root

With the temporary root feature you can temporarily set any object as the root object.
For instance, you can set the top object in a given subtree as the temporary root. This removes any objects above the selected temporary root, and facilitates navigation in that subtree.

Temporary Root sets the specified object as the current (temporary) root object. First, select an object and then choose **Temporary Root**. This displays the following dialog box, Figure 3-193.



Set Temporary Root

Filter object id: %

Filter object type: %

◆ Search tree
◇ Search list
◇ Search tree and list

Items

Selection: ++

Matched ids : 0

Figure 3-193. Set Temporary Root

This dialog box has the following fields:

Filter object id	Use this field to define a specific subset of objects to display, based on the object id. When you make an entry in this field, the dialog box displays all objects whose object ID matches the text string defined in this field
Filter object type	Use this field to define a specific subset of objects to display, based on the object type. When you make an entry in this field, the dialog box displays all objects whose object type matches the type specified in this field
Search tree Search list Search tree and list	Only one of those can be set. Search tree means display only tree objects. Search list means display only list object. Search tree and list means display both in the dialog box. This condition is also used with the filter pattern.

Items	Displays all objects which are matched with the above defined conditions.
Selection	Contains the selected object id and is used to set the temporary root object.
Matched ids	Counts the objects which are displayed after pressing the **Search** button.

If you do not select an object before you choose **View Temporary Root**, the Items and Selection fields are empty.

Filters are optional. You can use any combination of filter fields (one, both, or none) in conjunction with the Search buttons. When making an entry in a filter field, two characters can be used as wildcards. “%” sets the filter to match zero or more characters, and “_” sets the filter to match one character. Click on **Search** button and then the matched objects are displayed in the Items field. Select from that an object id.

If an object is selected and then choose **View Temporary Root** the Items are empty but the Selection field contains the object id of the selected object.

Click on **OK** when you are finished with this dialog. The object in the Selection field becomes the temporary root and gets a parent button. The next level of objects is displayed. Clicking the parent button once expands the presentation and displays the parent object above the temporary root, and its next level (direct children). Thus you can step your way back up to the original root object.

View - Project Root

Project Root resets the temporary root object. When you choose **Project Root** in the **View** menu, the actual project root object is set as the current root object and its next structure level is displayed. It is not necessary to select a specific object prior to choosing **Project Root**.

3.5.5.4 Object Menu

The **Object** menu is shown in Figure 3-194.

Object

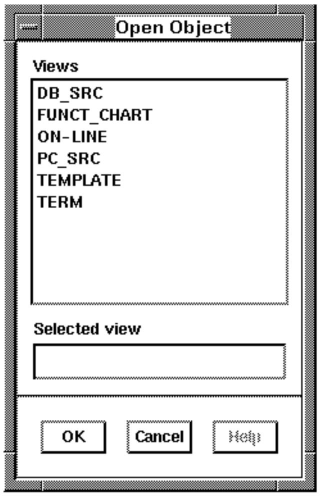
Open
Open Type
Print
Print Type
Information
Special Commands

Figure 3-194. Structure Builder - Object Menu

Object - Open

Open in the **Object** menu opens a a specific view of an object. To open an object, select the object, and then choose **Open** from the **Object** menu. Some objects have just one view, in which case, the view of the object is opened. For instance, if an object has just a template view, the Template Builder view of the object is opened so that you can view and edit object attributes.

Some objects have multiple views. When you choose **Open** for an object with multiple views, a dialog box such as the one shown below is displayed so that you can select the appropriate view, Figure 3-195.



Open Object

Views

DB_SRC
FUNCT_CHART
ON-LINE
PC_SRC
TEMPLATE
TERM

Selected view

Figure 3-195. View Selection Box

Select the view and then click on the **OK** button.

Object - Open Type

Open Type opens a multiple-object template form for all objects of a selected type in the selected subtree. First, select the parent object to specify the subtree. Then choose **Open Type** from the **Object** menu. This displays the Object Type dialog box, Figure 3-196. This dialog box displays all objects types; however, you must select an object that is a child of the selected parent.

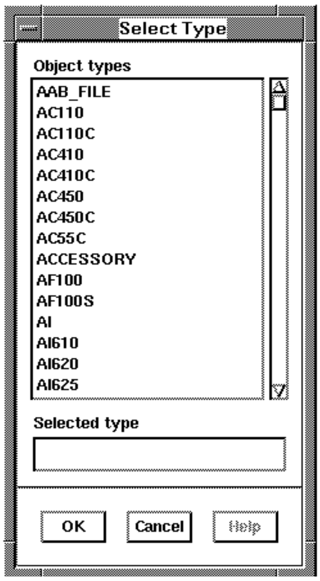
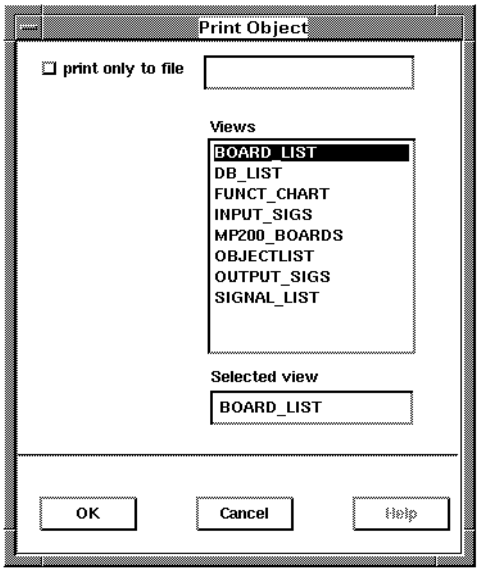


Figure 3-196. Open Type

Object - Print

Print prints a report for an object. The report is sent to the PostScript printer queue. To print a report, first select the object, and then choose **Print** from the **Object** menu. This displays the Print Object dialog box, Figure 3-197.



Selected view: BOARD_LIST

Figure 3-197. Print Object

This dialog box has the following fields:

print only to file	Use this field to create only a postscript file in the AdvaBuild user directory (default).
Views	Use this selection box to select the type of report, or template Builder screen dump.
Selected view	Contains the selected view which will be used.

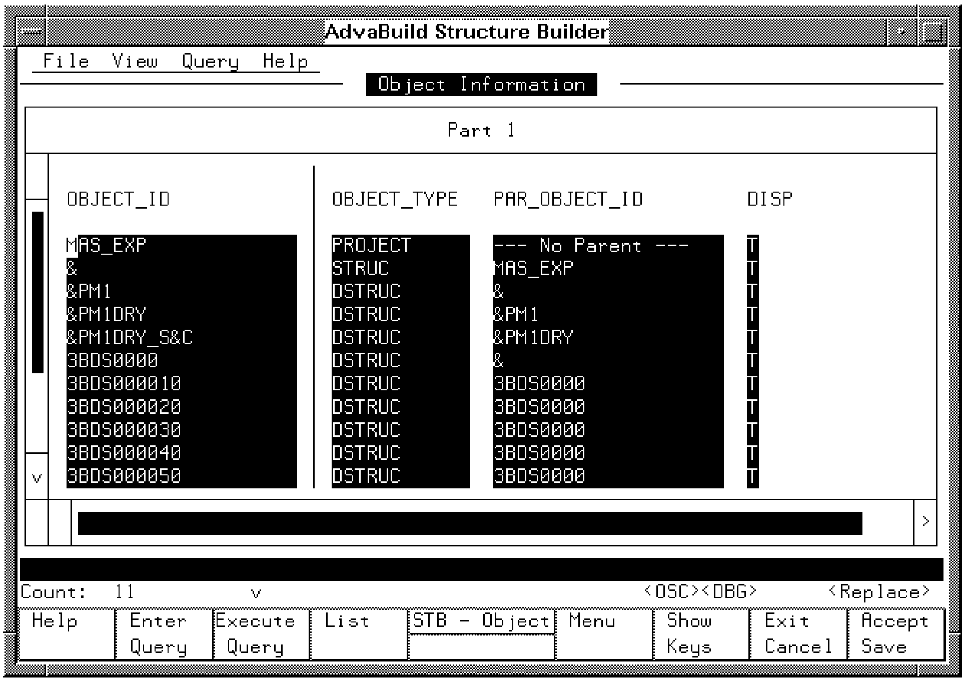
Click on **OK** when you are finished with this dialog.

Object - Print Type

Print Type is only applicable for object types that have this command defined for them. **Print Type** generates a report for multiple objects of a selected type (similar to **Open Object Type**). **Print Type** displays a dialog box for selecting the object type. You must select an object type that is a child of the selected parent object. After you select the object type, the Print Object dialog box is provided as described above under **Print**.

Object - Information

Information opens the Object Information Window for the selected object, Figure 3-198. This window provides the means to define and execute a database query that retrieves certain objects based on given a set of search keys (for example, parts of the object identification and object type).



```
OBJECT_ID          OBJECT_TYPE   PAR_OBJECT_ID          DISP

MAS_EXP            PROJECT       --- No Parent ---      T
&                  STRUC         MAS_EXP                T
&PM1               DSTRUC        &                      T
&PM1DRY            DSTRUC        &PM1                   T
&PM1DRY_S&C        DSTRUC        &PM1DRY                T
3BDS0000           DSTRUC        &                      T
3BDS000010         DSTRUC        3BDS0000               T
3BDS000020         DSTRUC        3BDS0000               T
3BDS000030         DSTRUC        3BDS0000               T
3BDS000040         DSTRUC        3BDS0000               T
3BDS000050         DSTRUC        3BDS0000               T

Count:  11         v                       <OSC><DBG>      <Replace>
```

Figure 3-198. Object Information Window

Object - Special Commands

Special commands provides functions for compiling, installing, and downloading the MOD 300 database, for transferring database files between different MOD 300 database configuration platforms, and other miscellaneous functions. **Special commands** displays a selection box with the applicable special commands, depending upon the selected object type, Figure 3-199.

NOTE

To use the following special commands: COMPILE, INSTALL, DECOMPILE, EXPORT_REFS, and INC_DLOAD, you must have Admin or Config authority.

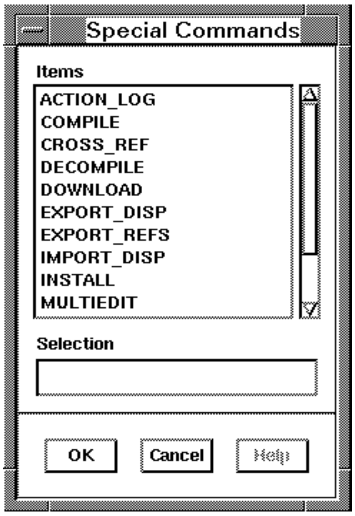


Figure 3-199. Special Commands

This dialog box has the following fields:

Items	Use this selection box to select the required special command.
Selection	This field shows the selected special command.

The special commands provided in this dialog box vary depending upon the object that you select before choosing **Special Commands** from the **Object** menu.

Some object types require functions other than those provided via the menu bar. For example, tools for handling AMPL source code. Such functions are available under the **Special Commands** menu item when the appropriate object is selected. A selector box is provided with the defined special commands to select and execute one. The commands have to be defined in the data base.

3.5.5.5 Link Menu

The **Link** menu is shown in Figure 3-200.

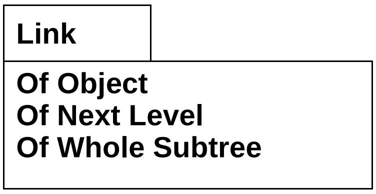
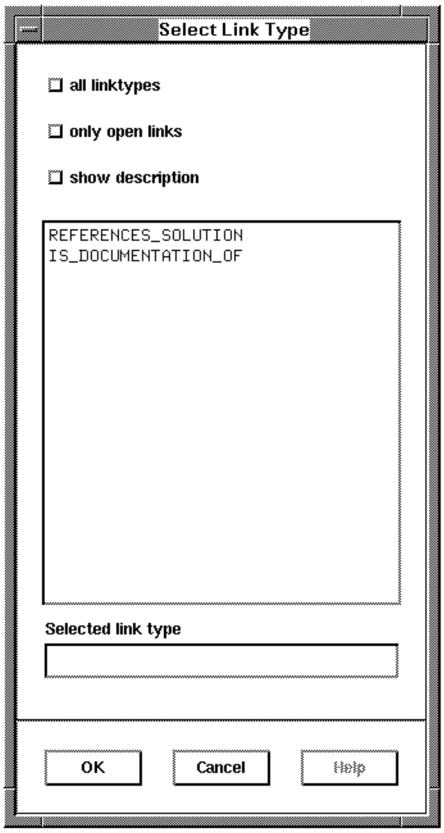


Figure 3-200. Structure Builder - Link Menu

When you choose one of the **Link** menu items the following dialog box is displayed, Figure 3-201.



☐ all linktypes

☐ only open links

☐ show description

```
REFERENCES_SOLUTION
IS_DOCUMENTATION_OF
```

Selected link type

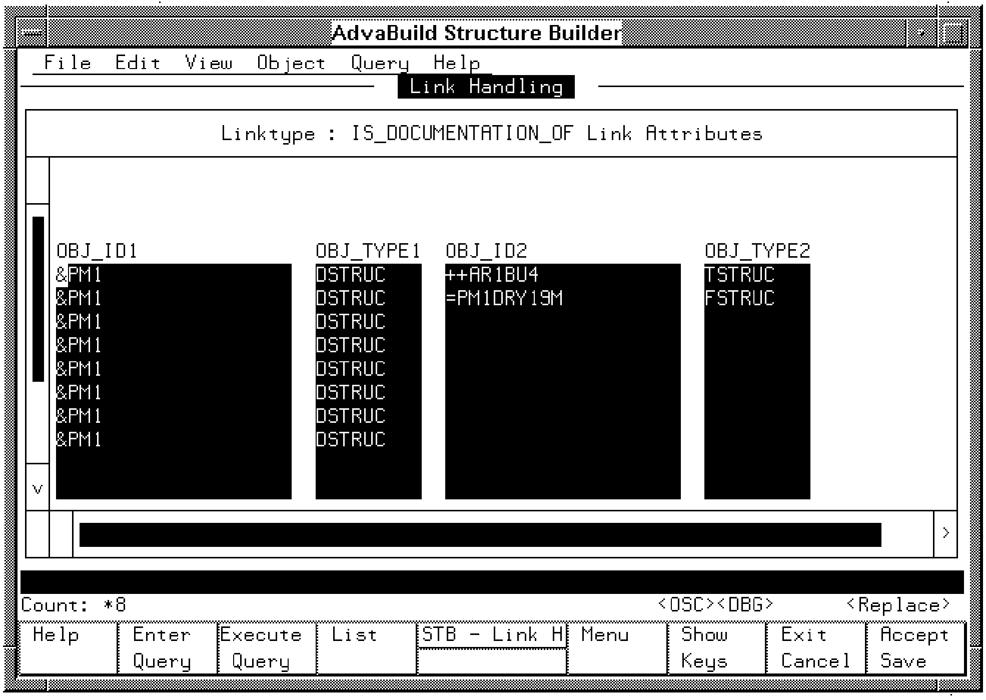
Figure 3-201. General Link handling Window

This dialog box has the following fields:

all linktypes	Use this check button to display all link types in the link builder.
only open links	Use this check button to display only open links in the link builder.
show description	Use this check button to display a description of the link type in the current dialog box.
Selected link type	This field shows the selected link type to be used in the link builder.

Link - Of Object

Of Object allows you to create, delete, and modify links of the selected object. When you select the link type in the dialog box, the following window is displayed, Figure 3-202.



Linktype : IS_DOCUMENTATION_OF Link Attributes

OBJ_ID1	OBJ_TYPE1	OBJ_ID2	OBJ_TYPE2
&PM1	DSTRUC	++AR1BU4	TSTRUC
&PM1	DSTRUC	=PM1DRY19M	FSTRUC
&PM1	DSTRUC		
&PM1	DSTRUC		
&PM1	DSTRUC		
&PM1	DSTRUC		
&PM1	DSTRUC		
&PM1	DSTRUC		

Figure 3-202. Link-Of-Object Window

Link - Of Next Level

Of Next Level allows you to create, delete, and modify links of the next level of the selected object. When you select the link type in the dialog box, the following window is displayed, Figure 3-203.

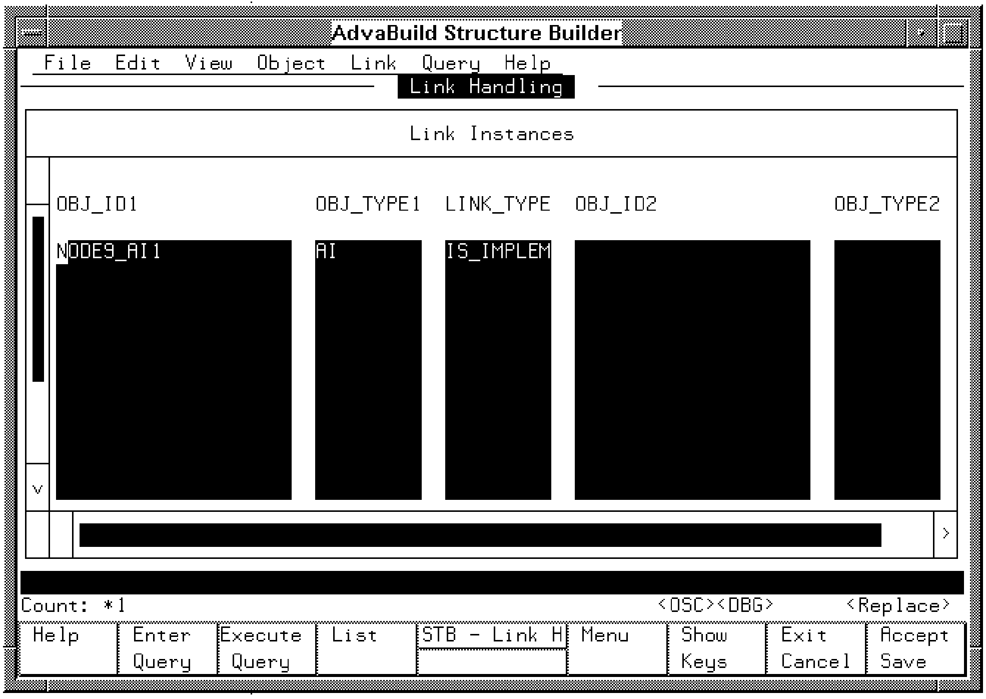
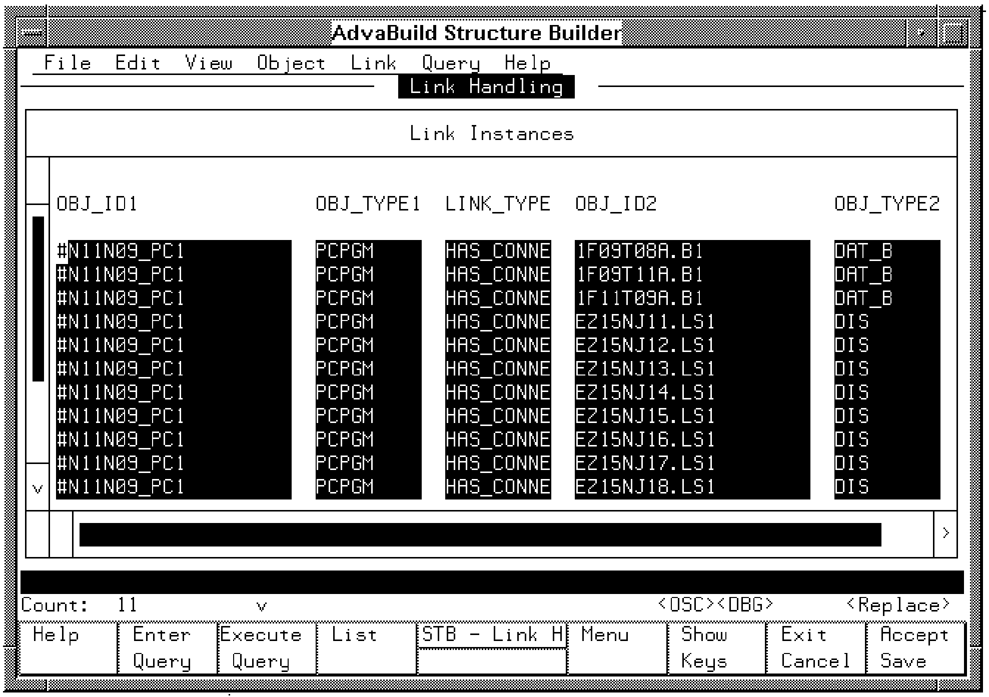


Figure 3-203. Link-Of-Next-Level Window

Link - Of Whole Subtree

Of Whole Subtree allows you to create, delete, and modify links of whole subtree of the selected object. When you select the link type in the dialog, the following window is displayed, Figure 3-204.



OBJ_ID1	OBJ_TYPE1	LINK_TYPE	OBJ_ID2	OBJ_TYPE2
#N11N09_PC1	PCPGM	HAS_CONNE	1F09T08A.B1	DAT_B
#N11N09_PC1	PCPGM	HAS_CONNE	1F09T11A.B1	DAT_B
#N11N09_PC1	PCPGM	HAS_CONNE	1F11T09A.B1	DAT_B
#N11N09_PC1	PCPGM	HAS_CONNE	EZ15NJ11.LS1	DIS
#N11N09_PC1	PCPGM	HAS_CONNE	EZ15NJ12.LS1	DIS
#N11N09_PC1	PCPGM	HAS_CONNE	EZ15NJ13.LS1	DIS
#N11N09_PC1	PCPGM	HAS_CONNE	EZ15NJ14.LS1	DIS
#N11N09_PC1	PCPGM	HAS_CONNE	EZ15NJ15.LS1	DIS
#N11N09_PC1	PCPGM	HAS_CONNE	EZ15NJ16.LS1	DIS
#N11N09_PC1	PCPGM	HAS_CONNE	EZ15NJ17.LS1	DIS
#N11N09_PC1	PCPGM	HAS_CONNE	EZ15NJ18.LS1	DIS

Count: 11 v <OSC><DBG> <Replace>

Figure 3-204. Link-Of-Whole-subtree Window

3.5.5.6 Query Menu

Query Enter (only in list and aspect window)

This menu item is used to execute a data base query. To execute a query, select **Query Enter**. This provides a dialog box for specifying the query parameters, Figure 3-205.

Specify whether you want to query just children of the specified parent object, or the whole sub-tree. If the radio button *Parent Object* is set (default condition), the list window will be filled only with direct child objects of the selected parent object. If the radio button *Whole subtree* is set, the list window will be filled with the whole sub-tree of the selected parent object.

There are two filters (Object ID and Object Type) which can be used to specify a sub-set of the expected list objects. The percent sign "%" and the underscore "_" are used as wildcards. "%" sets the filter to match zero or more characters, and "_" sets the filter to match exactly one character. Click on the **OK** button to create a list window with the specified list objects.

Note it takes time for querying a large number of list objects and to refill the list window.

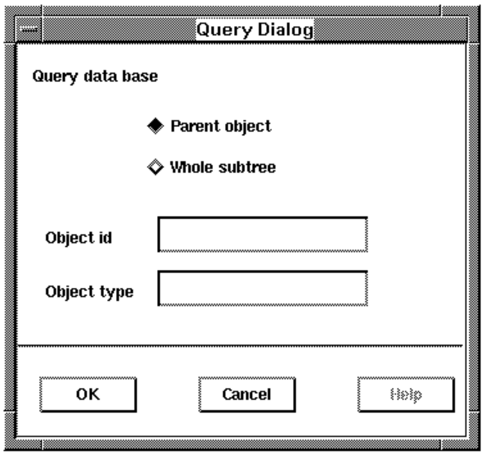


Figure 3-205. Query Dialog Box

This dialog box has the following fields:

Parent object	Use this toggle button to search objects of next level.
Whole subtree	Use this toggle button to search objects of whole subtree.
Object id	Use this filter to specify objects by object id.
Object type	Use this filter to specify objects by object type.

3.5.5.7 Utility Menu

The **Utility** menu is shown in Figure 3-206.

Utility	
Configure > **Plant Documentation Package** **File Manager**	**Plant Documentation Builder** **Diagram Builder** **Reports**

Figure 3-206. Structure Builder - Utility Menu

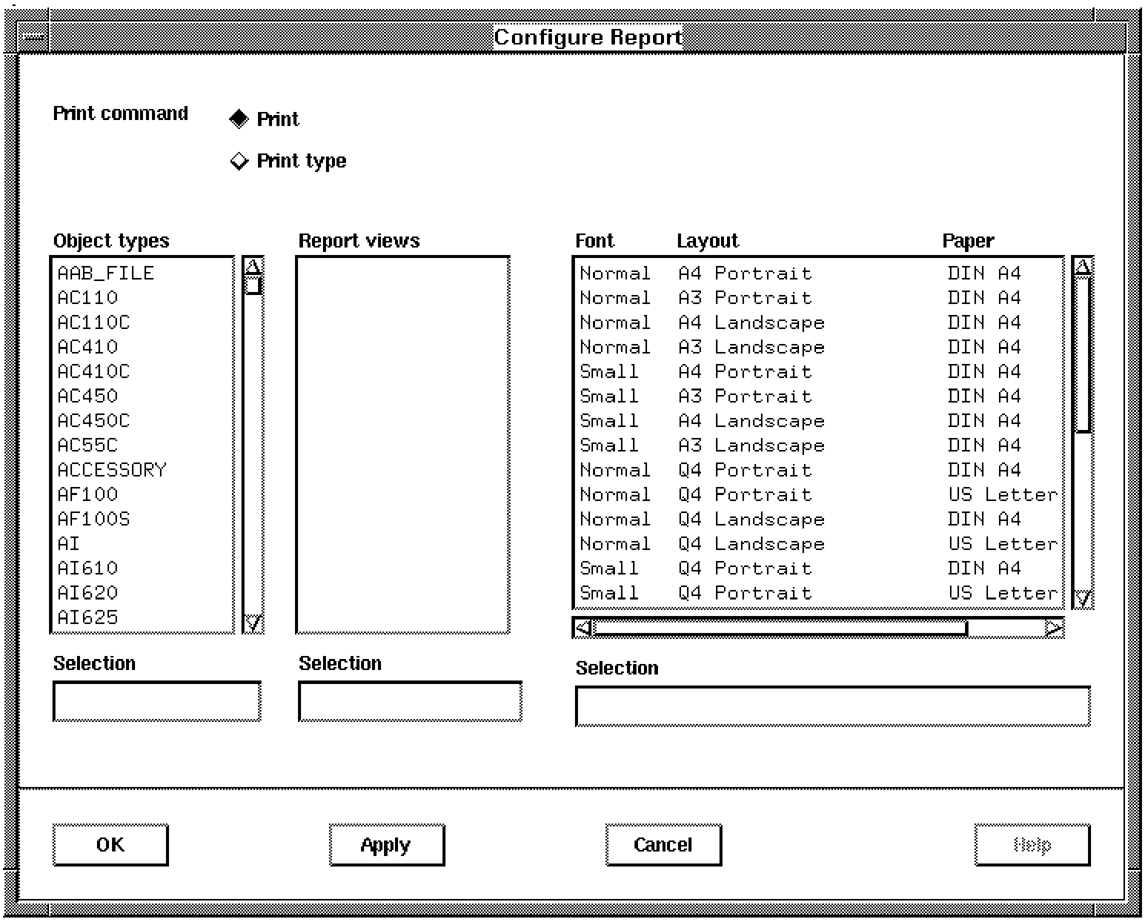
Utility - Configure

Configure displays a menu with three choices: **Plant Documentation Builder**, **Diagram Builder**, and **Repor**ts. Select either one of these choices to access the tool for configuring one of these builders. These configure functions are only available for CONFIG users. These menu items are dimmed for other users.

Configure Reports

This menu item provides access to the ORACLE SQL Report Writer.

To access the Report Writer, select an object and then select **Configure Report Writer**. This displays the following dialog box, Figure 3-207.



Print command: ◆ Print ◇ Print type

Object types:

AAB_FILE
AC110
AC110C
AC410
AC410C
AC450
AC450C
AC55C
ACCESSORY
AF100
AF100S
AI
AI610
AI620
AI625

Font	Layout	Paper
Normal	A4 Portrait	DIN A4
Normal	A3 Portrait	DIN A4
Normal	A4 Landscape	DIN A4
Normal	A3 Landscape	DIN A4
Small	A4 Portrait	DIN A4
Small	A3 Portrait	DIN A4
Small	A4 Landscape	DIN A4
Small	A3 Landscape	DIN A4
Normal	Q4 Portrait	DIN A4
Normal	Q4 Portrait	US Letter
Normal	Q4 Landscape	DIN A4
Normal	Q4 Landscape	US Letter
Small	Q4 Portrait	DIN A4
Small	Q4 Portrait	US Letter

Figure 3-207. Configure Reports Dialog Box

Choose either the Print or Print Type toggle button. Select an object type, then one of the report views and a layout format that are displayed for the selected object type. Press the **OK** button or **Apply** button to invoke the SQL*Report Writer. By clicking the **Apply** button the configure report writer dialog box will not be removed so you can continue to define another report.

Utility - Plant Documentation Builder

Plant Documentation Builder displays the DOB_Generation window for printing documentation packages.

Utility - File Manager

File Manager displays the HP File Manager, Figure 3-208. The default directory is the current working directory. The VUE PAD editor can be called from the HP File Manager menu. To edit a file, select the file name in the file manager window, and then choose **Edit** from the **actions** menu. To quit the HP File Manager, choose **Exit** from the **File** menu. For more information refer to the HP File Manager user's manual.

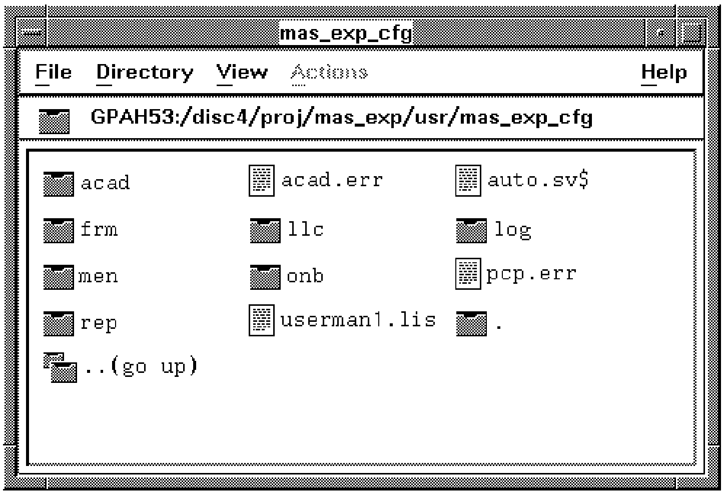


Figure 3-208. File Manager Window

Choose **Exit** from the **File** menu to quit the VUE PAD editor. For more information refer to the VUE PAD editor user's manual.

3.5.5.8 Help Menu

The **Help** menu is shown in Figure 3-209.

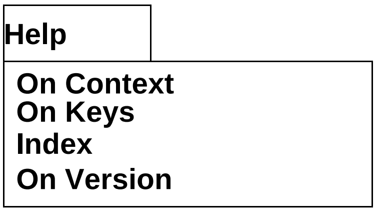


Figure 3-209. Help Menu

Help - On Context

This menu item is not supported yet and grayed out.

Help - On Keys

This menu item is not supported yet and grayed out.

Help - Index

Index displays on mouse cursor's position a pop up menu with a list of available help topics. Navigate to the desired topic and then select it. A separate tool displays the desired help topic.

How to use the *Advant OCS Users Documentation CD* via Help-Index

To use the *Advant OCS Users Documentation CD* you must first mount the CD ROM. After you have connected the CD ROM hardware to the AS500ES and inserted the *Advant OCS Users Documentation CD*, login as user root and mount the new device in the following way:
Open a hpterm window and enter:

```
cd /
mount -v -t cdfs /dev/dsk/c201d1s0 /UPDATE_CDROM/
```

(here /dev/dsk/c201d1s0 is used for CD ROM address SCSI1, for address SCSIx use /dev/dsk/c201dxs0)

Help - On Version

On Version shows the Version information of the software that is running.

3.5.6 Menus for Structure Builder Object Information

3.5.6.1 File Menu

The **File** menu provides commands for saving entries made from the work space of the form to the database, for generating a printout and exiting the Object Information form, Figure 3-210.

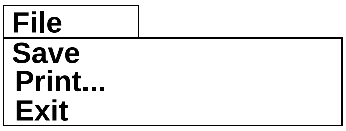


Figure 3-210. Object Information - File menu

File - Save

Save saves entries made during the current editing session from the work space of the form to the database. All modifications made since the last Save are stored in the database. You can also use **Accept/Save** (**<f8>**) to perform the **Save** function.

The Object Information form prompts you when the function is done:

1 Transaction complete -- n records posted and committed

If an error occurs during the save, the whole transaction is rolled back and the cursor is positioned on the row containing the first object which causes an error.

File - Print

Print writes all objects displayed in the form (according to the last query condition) to a file for printing. You can select the attributes to be printed from the Select Attributes selection box. Note that the names listed in the selection box are the display names in the form. These names are not necessarily the attribute names in the database. You can choose to either print out the attributes or print only to file via the File Print dialog box. Note that the order of attributes in the print out or file is determined by the last execution of the Order-By selection dialog (**View - Order** menu item), or the default order when you invoke the Object Information window.

File - Exit

Exit closes the Object Information window (exit the form). This command is equivalent to selecting **Exit/Cancel**. If you modify data in the form and attempt to exit without saving, an alert box is displayed. This box prompts you to save the changes or cancel the operation.

Navigate to the field you want and press **Select** to activate it.**Yes** saves the pending modifications into the database. **No** discards the modification and exits the form. **Cancel** returns to where you were in the form without saving the changes.

3.5.6.2 View Menu

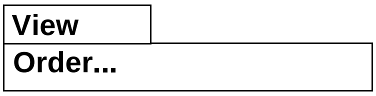


Figure 3-211. Object Information - View menu

View - Order

Order provides the means to redefine the sequence of objects displayed in the form. By default, objects are ordered by the values of their internal **tree_id** (c_tree_id) in ascending order. **Order** displays the Order By dialog box. Use this box to assign numbers between 1 and 9 to the attributes. The objects are ordered by the value of the attribute with the lowest number first. Objects with the same value are ordered by the value of the attribute with the next highest number and so on. Holes (gaps) in the numbering are allowed, but a number may not be assigned to two attributes. Navigate to the attribute names and enter values between 1 and 9 into the corresponding fields of the dialog box (left of the attribute name).

3.5.6.3 Query Menu

Refer to the description of the **query** menu items in Section 3.5.1.1, Query Menu.

3.5.7 Menus for Solution Aspects Window

The menus for the Solution Aspects window provide a subset of the functionality provided by the equivalent menus on the Structure Builder menu bar (List Objects window), Table 3-20.

Table 3-20. Solution Aspects Menu Items

Menu	Menu item
File	Print
	Close
Edit	Insert
	Rename
	Delete
Object	Open
	Print
	Information
	Special Commands
Link	Of Object
	Of Next Level
	Of Whole Subtree
	Type
Query	Enter

The following sections describe the differences to the Structure Builder Menu Bar (List Objects Window).

Edit / Insert

When you choose **Insert** from the **Edit** menu the following dialog box is displayed, Figure 3-212.

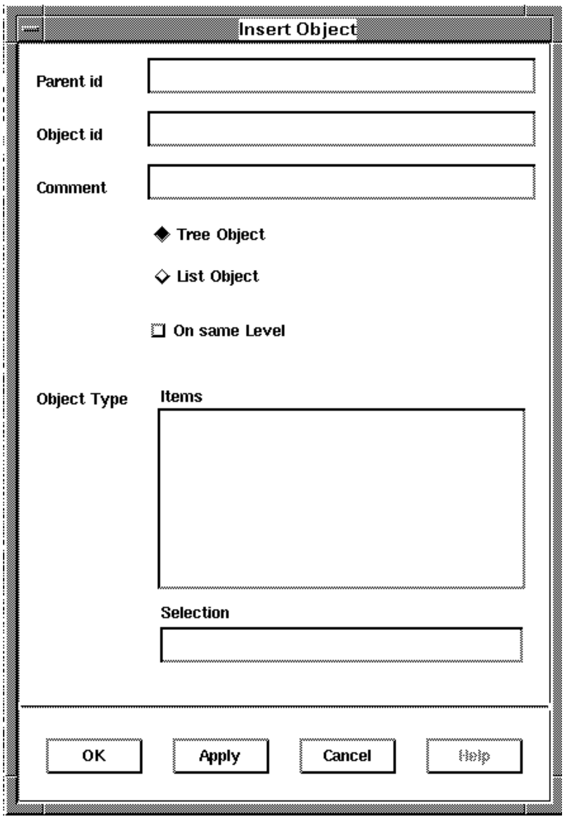


Figure 3-212. Insert Object Dialog Box, Solution Aspects Window

Because a Solution's aspect window does not display objects belonging to a dedicated parent object; but rather, displays objects residing somewhere in the AS 500ES main structures, you have to specify the required predecessor of the Solution aspect object you want to insert. Enter the object ID for the predecessor in the Parent id field. Enter the required Solution aspect object's ID in the Object id field.

If you de-select the On same level toggle button (default), the Solution aspect object is inserted one level deeper in the object-hierarchy compared with the specified predecessor. The specified predecessor acts as parent object of the Solution aspect object. If you select the On same level toggle button, the Solution aspect object is inserted on the same level in the object-hierarchy, directly below the specified predecessor.

If you select the Tree Object toggle button (default) the Solution aspect object is inserted as a tree object. If you select the List Object toggle button, the Solution aspect object is inserted as a list object.

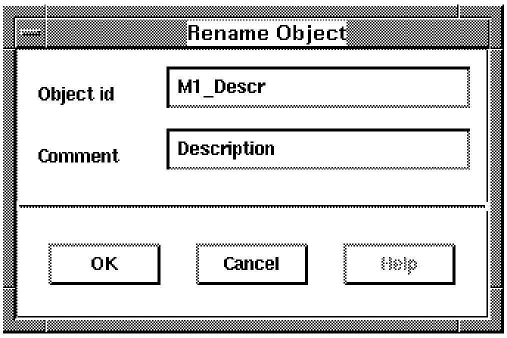
The Object Type list displays all object types allowed for the Solution aspect object. The displayed types depend on the selected predecessor (Parent id field) as well as on the On same level toggle button's state. If the Parent id field is empty, no types are displayed.

To determine the required Solution aspect object's type, select one in the Object Type list. Your selection is indicated in the Selection field. As an alternative, you can manually insert the object type in this field.

Click on **OK** to accept your entries in the dialog box, and then close the box. As an alternative, you can use **Apply** rather than **OK**. This accepts your current entries and leaves the dialog box open so that you can insert additional Solution aspect objects. Click on **OK** when you are finished with the dialog box. Use **Cancel** to close the dialog box without accepting the entries.

Edit / Rename

To rename a Solution aspect object, select the required object and choose **Rename** item from the **Edit** menu. This displays the following dialog box, Figure 3-213.



Object id: M1_Descr
Comment: Description

Figure 3-213. Rename Object Dialog Box, Solution Aspects Window

The Solution aspect object's ID and comment are displayed in the Object id and Comment fields respectively. You can modify them there.

Click on **OK** to accept your entries in the dialog box, and then close the box.

Use **Cancel** to close the dialog box and do not accept the entries.

NOTE

By choosing **Rename** from the **Edit** menu in a Solution's Solution Aspects window, you can only rename a dedicated Solution aspect object, but you can not include its subtree (if one exists) into the rename procedure. For renaming an object together with its subtree you must choose **Rename** from the **Edit** menu in either in the tree representation's window or in the List window of the Structure Builder.

Query / Enter

To limit the scope of objects represented in a Solution Aspects window, select **Enter** from the **Query** menu. This displays the following dialog box, Figure 3-214.

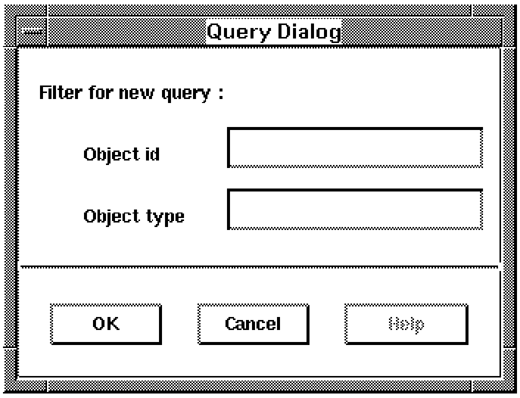


Figure 3-214. Query Dialog Box, Solution Aspects Window

You can specify an Object ID and/or Object type filter to limit the scope of Solution aspect objects listed in the Solution Aspects window.% (arbitrary text string) and _ (arbitrary character) are wild cards and can be used in combination. Object ID and Object type filters can also be used in combination.

Click on **OK** to accept your entries in the dialog box, and then close the box.

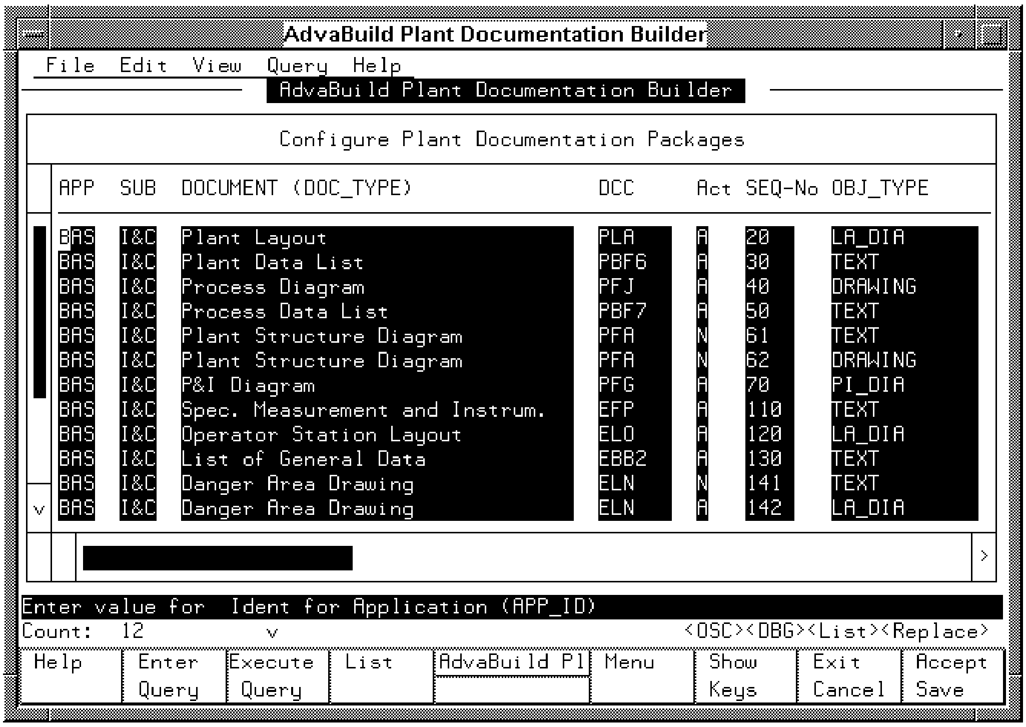
Use **Cancel** to close the dialog box and do not accept the entries.

NOTE

You can use the Query Dialog box to refresh the Solution Aspects window's content. Just leave the Object id and Object type fields empty and click on **OK**.

3.5.8 Menus for Plant Documentation Builder Configure Window

This section describes the menu items for the PDB Configure window, Figure 3-215.



Configure Plant Documentation Packages

APP	SUB	DOCUMENT (DOC_TYPE)	DCC	Act	SEQ-No	OBJ_TYPE
BAS	I&C	Plant Layout	PLA	A	20	LA_DIA
BAS	I&C	Plant Data List	PBF6	A	30	TEXT
BAS	I&C	Process Diagram	PFJ	A	40	DRAWING
BAS	I&C	Process Data List	PBF7	A	50	TEXT
BAS	I&C	Plant Structure Diagram	PFA	N	61	TEXT
BAS	I&C	Plant Structure Diagram	PFA	N	62	DRAWING
BAS	I&C	P&I Diagram	PFG	A	70	PI_DIA
BAS	I&C	Spec. Measurement and Instrum.	EFP	A	110	TEXT
BAS	I&C	Operator Station Layout	ELO	A	120	LA_DIA
BAS	I&C	List of General Data	EBB2	A	130	TEXT
BAS	I&C	Danger Area Drawing	ELN	N	141	TEXT
BAS	I&C	Danger Area Drawing	ELN	A	142	LA_DIA

```
Enter value for  Ident for Application (APP_ID)
Count:  12         v                           <OSC><DBG><List><Replace>
```

Figure 3-215. PDB Configure Window

3.5.8.1 File Menu

The File menu for the Plant Documentation Builder is shown in Figure 3-216.

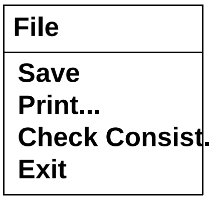


Figure 3-216. Plant Documentation Builder - File Menu

File - Save

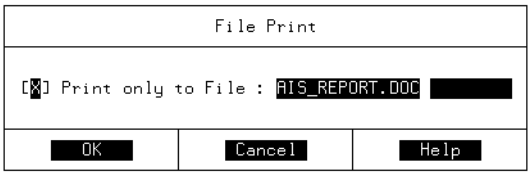
Save saves modified entries of the standard Plant Documentation Package to the DOC_RELATION table. You can also use **Accept/Save** (**<f8>**) to perform the **Save** function.

The following message is displayed when the save is done:

> " Transaction complete -- n records posted and committed."
> or
> "No changes to commit."

File - Print

Print stores the current configuration of the Plant Documentation package (as defined in the PDB_Configure window) into a postscript file and then prints the file. When you choose **Print** from the **File** menu, the Print File dialog box is displayed, Figure 3-217.



File Print

[☒] Print only to File : AIS_REPORT.DOC

Figure 3-217. Print File Dialog Box for PDB Configuration Window

You can select whether the printout should just be stored in the file, or be printed as well.

The file is given a ".ps" extension. Therefore, do not assign the file an extension when you define the filename in the Print File dialog box. The filename can be defined with directory information (directory must exist).

- direct like /users/my_name/my_docs/<filename>
- or relative like ./tmp/<filename>

NOTE

Environment-variables cannot be used.

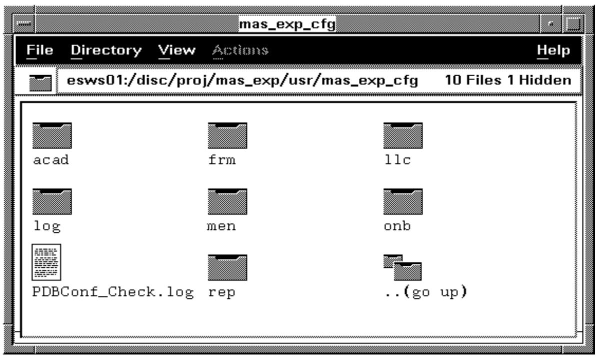
If the filename is given without path information the predefined Spoolfile is placed in the Project Users Subdirectory /proj/<Project Name>/usr/<User Name> (e.g. for Project Name: mas_exp,, for User Name: mas_exp_cfg).

The execution of print to file can be checked by choosing **File Manager** from the **Utility** menu in STB.

File - Check Consits.

Check Consistency checks the actual state of PDB Configuration and writes results into the file /proj/<Project Name>/usr/<User Name>/PDBConf_Check.log. This function takes a while, so be patient.

You can access the file using the File Manager menu item in the Utility menu. An example is shown in Figure 3-218.



esws01:/disc/proj/mas_exp/usr/mas_exp_cfg 10 Files 1 Hidden

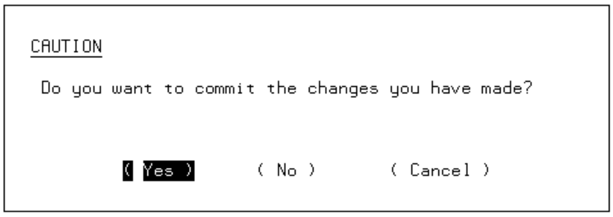
Figure 3-218. File Manager Window

Parts of check are:

- DOC_RELATION-Pointer to table OBJ_COMMAND (via OBJ_TYPE, COMMAND, COM_VIEW) without definition in table OBJ_COMMAND. The Document Types detected by these consistency checks cannot be activated during Package Generation.
- Inconsistency of Attribute >DOC_TYPE< in tables TEXT, DRAWING, HOOKUP, LA_DIA, CI_DIA and PI_DIA and DISP. Used Document Types do not correspond with the related List_of_Values. (TEB inputs in AdvaBuild Release 2.1-0 for the listed object types are checked.)
- A list with allowed entries for the attribute DOC_TYPE for objects of types TEXT, DRAWING, HOOKUP, LA_DIA, CI_DIA and PI_DIA is shown.

File - Exit

Exit closes the PDB Configure window. This command is equivalent to selecting **Exit/Cancel**. If you modify data in the form and attempt to exit without saving, an alert box is displayed, Figure 3-219. This box prompts you to save the changes or cancel the operation.



CAUTION

Do you want to commit the changes you have made?

Figure 3-219. Commit Alert Box

Navigate to the field you want and press **Select** to activate it.**Yes** saves the pending modifications into the database. **No** discards the modification and exits the form. **Cancel** returns to where you were in the form without saving the changes.

3.5.8.2 Edit Menu

The **Edit** menu is only available in the PDB_Configure window and is shown in Figure 3-220.

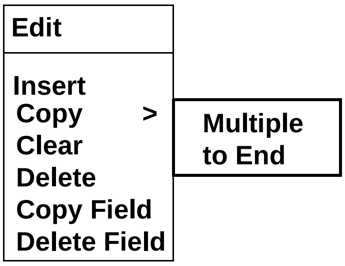


Figure 3-220. Plant Documentation Builder - Edit Menu

Edit - Insert

This item is for inserting a new document type (or a new kind of document for a document type) in the DOC_RELATION table.

Insert creates a new (empty) record in the DOC_REALTION table below the current cursor position. Use <**Tab**> and <Shift-**Tab**> to navigate the fields for the new record. Refer to Section 3.4.5.3, How to Configure the DOC_RELATION Table for information on how to configure these fields.

Edit - Copy

Copy makes single or multiple copies of a selected record, or it can make one copy of each record in the DOC_RELATION table starting at a selected record. **Copy** displays a pop-up menu with the following options: **Multiple** and **To End**.

Edit-Copy-Multiple

Multiple creates *n* number of copies of the selected record. To make multiple copies of a record, first position the cursor on the line for the record that you want to copy, and then choose **Edit-Copy-Multiple**. This displays the Number of Copies dialog box. Enter the number of copies that you want to make, and then select **Accept/Save**.

Edit - Copy - To End

This function is not available.

Edit - Clear

This clears the previous **Insert** action and/or field-modifications in the current record if an **Accept/Save** has not been performed. This allows you to rollback after incorrect data has been entered but not yet been saved.

Edit - Delete

This deletes individual records from the table. The cursor must first be positioned on the record to be deleted by using the up and down arrow keys. All changes are saved if **Accept/Save** is selected.

If a record is deleted by mistake, all changes can be cancelled by selecting **Exit/cancel** prior to saving. This displays following message:

```
CAUTION
Do you want to commit the changes you have made?
```

To cancel the delete command, click on **No** and then press <**Return**>. The delete action is cancelled, but note that all changes made since the last save will be lost.

Edit - Copy Field

Copy Field copies the value from the field of the previous record, and writes the value into the corresponding field where the cursor is positioned.

Edit - Delete Field

Delete Field deletes the contents of the selected field.

3.5.8.3 View Menu

The **View** menu for the Configuration of PDB is shown in Figure 3-221.

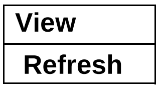


Figure 3-221. Plant Documentation Builder - View Menu

View - Refresh

This refreshes and redraws the Plant Documentation Builder window.

3.5.8.4 Query Menu

Refer to description of **Query** menu in {Section 3.5.1.1, Query Menu}.

3.5.8.5 Help Menu

Refer to description of **Help** menu in {Section 3.5.1.2, Help Menu}.

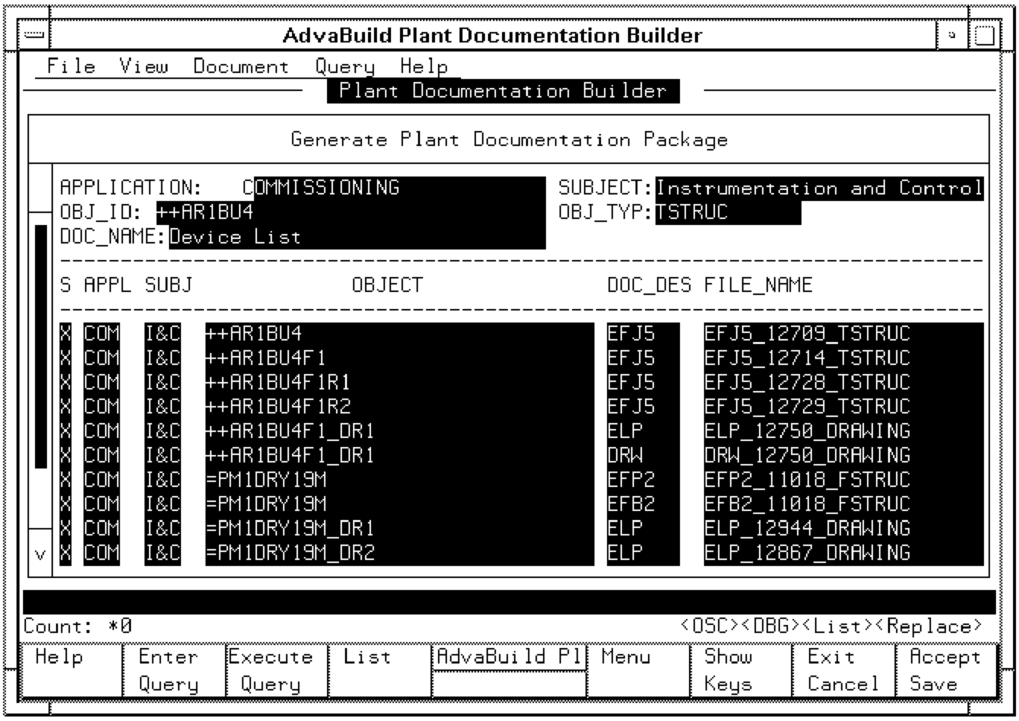
3.5.8.6 Special Keys

For direct access to a definite record the following key-combinations are installed:

ESC / ESC 0	Go to first record
ESC / ESC 1	Go to last record
ESC / ESC 2	Go to record number <no> (Input of <no> via Input Box).

3.5.9 Menus for the Plant Documentation Builder Generation Window

This section describes the menu items for the PDB Package Generation Window, Figure 3-222.


AdvaBuild Plant Documentation Builder
File View Document Query Help
Plant Documentation Builder
Generate Plant Documentation Package
Help
Enter Query
Execute Query
List
AdvaBuild Pl
Menu
Show Keys
Exit Cancel
Accept Save

```
APPLICATION:   COMMISSIONING                 SUBJECT:Instrumentation and Control
OBJ_ID: ++AR1BU4                             OBJ_TYP:TSTRUC
DOC_NAME:Device List
---------------------------------------------------------------------------
S APPL SUBJ             OBJECT               DOC_DES FILE_NAME
---------------------------------------------------------------------------
X COM  I&C  ++AR1BU4                         EFJ5    EFJ5_12709_TSTRUC
X COM  I&C  ++AR1BU4F1                       EFJ5    EFJ5_12714_TSTRUC
X COM  I&C  ++AR1BU4F1R1                     EFJ5    EFJ5_12728_TSTRUC
X COM  I&C  ++AR1BU4F1R2                     EFJ5    EFJ5_12729_TSTRUC
X COM  I&C  ++AR1BU4F1_DR1                   ELP     ELP_12750_DRAWING
X COM  I&C  ++AR1BU4F1_DR1                   DRW     DRW_12750_DRAWING
X COM  I&C  =PM1DRY19M                       EFP2    EFP2_11018_FSTRUC
X COM  I&C  =PM1DRY19M                       EFB2    EFB2_11018_FSTRUC
X COM  I&C  =PM1DRY19M_DR1                   ELP     ELP_12944_DRAWING
X COM  I&C  =PM1DRY19M_DR2                   ELP     ELP_12867_DRAWING

Count: *0                                         <OSC><DBG><List><Replace>
```

Figure 3-222. PDB Package Generation Window

3.5.9.1 File Menu

The File menu for the Plant Documentation Builder is shown in Figure 3-223.

File

Print
Exit

Figure 3-223. Plant Documentation Builder - File Menu

File - Save

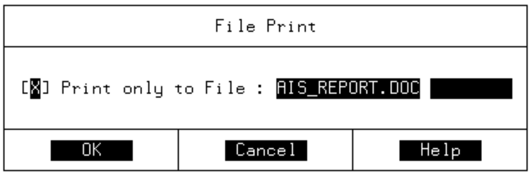
Save saves modified entries of the actual order selection to the table PRE_ORDER. You can also use **Accept/Save** (**<f8>**) to perform the **Save** function.

The following message is displayed when the save is done:

> " Transaction complete -- n records posted and committed."
> or
> "No changes to commit.".

File - Print

Print stores the actual list of documents to be printed (as defined on the PDB_Package_Generation window) into a postscript file and then prints the file. When you choose **Print** from the **File** menu, the Print File dialog box is displayed, Figure 3-224.



[☒] Print only to File : AIS_REPORT.DOC

Figure 3-224. Print File Dialog Box for PDB Generation Package Window

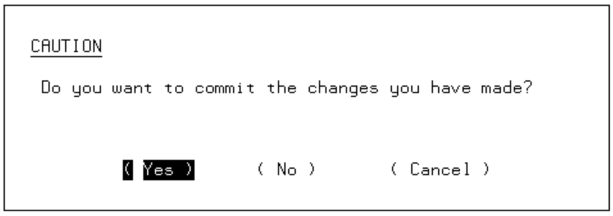
You can select whether the printout should just be stored in the file, or be printed as well.

The file is given a “.ps” extension. Therefore, do not assign the file an extension when you define the filename in the Print File dialog box.

If the filename is given without path information the predefined Spoolfile is placed in the Project Users Subdirectory /proj/<Project Name>/usr/<User Name> (e.g. for Project Name: mas_exp, for User Name: mas_exp_cfg).

File - Exit

Exit closes the Plant Documentation Builder window. This command is equivalent to selecting **Exit/Cancel**. If you modify data in the form and attempt to exit without saving, an alert box is displayed, Figure 3-225. This box prompts you to save the changes or cancel the operation.



CAUTION

Do you want to commit the changes you have made?

Figure 3-225. Commit Alert Box

Navigate to the field you want and press **Select** to activate it.**Yes** saves the pending modifications into the database. **No** discards the modification and exits the form. **Cancel** returns to where you were in the form without saving the changes.

3.5.9.2 View Menu

The **View** menu for the Plant Documentation Builder is shown in Figure 3-226.

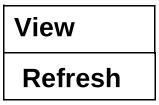


Figure 3-226. Plant Documentation Builder - View Menu

View - Refresh

This refreshes and redraws the Plant Documentation Builder window.

3.5.9.3 Document Menu

The Document menu provides the means to print a complete documentation package. This menu is only available in the PDB_Package_Generation window, and is shown in Figure 3-227.

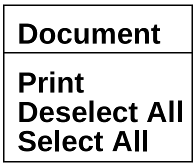
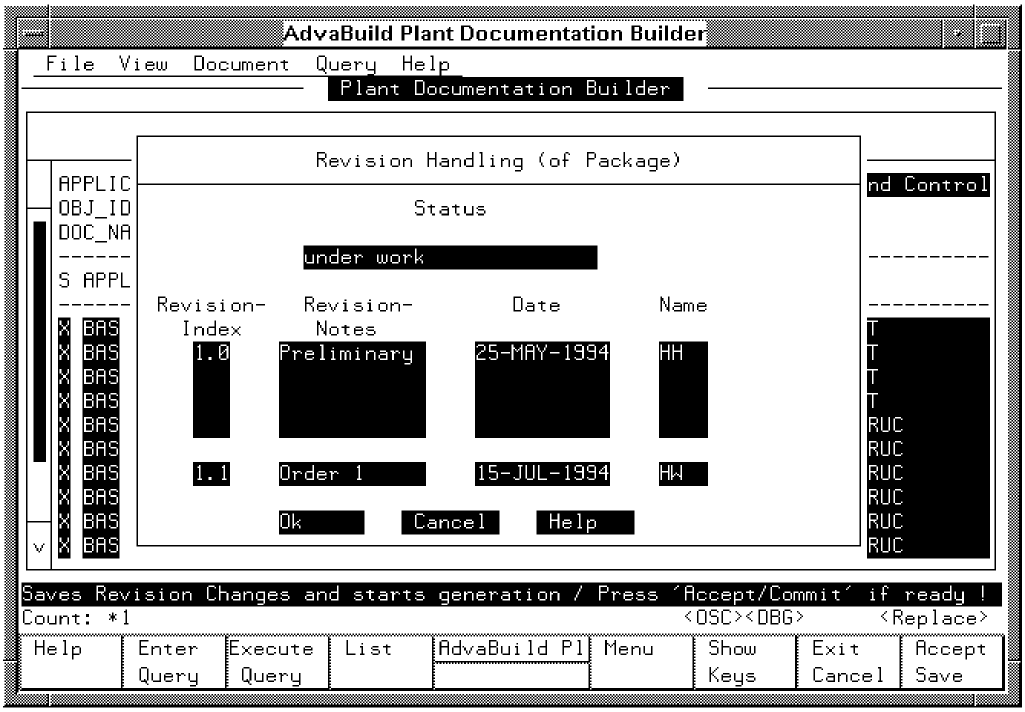


Figure 3-227. Plant Documentation Builder - Document Menu

Document - Print

This displays a revision handling dialog box for the selected documentation package, and the print spool window for printing the document package, Figure 3-228.



Revision Handling (of Package)

Status: under work

Revision-Index	Revision-Notes	Date	Name
1.0	Preliminary	25-MAY-1994	HH
1.1	Order 1	15-JUL-1994	HW

```
Saves Revision Changes and starts generation / Press 'Accept/Commit' if ready !
Count: *1                                   <OSC><DBG>      <Replace>
```

Figure 3-228. Revision Handling and Print Spool Window

Document - Deselect All

This deselects all selected documents in the PDB_Generation window (deselected documents are indicated by absence of **X** in S (Select) column). This function is useful when there are many selected documents in the PDB_Package_Generation window and you only want to select a few for printing.

Document - Select All

This selects all documents in the PDB_Package_Generation window (selected documents are indicated by **X** in S (Select) column).

3.5.9.4 Query Menu

Refer to description of **Query** menu in Section 3.5.1.1, Query Menu.

3.5.9.5 Help Menu

Refer to description of **Help** menu in Section 3.5.1.2, Help Menu.

3.5.10 Link Handler Menus

General

This section provides a summary of the link handling menu actions. The Link Handler has two windows:

- The General Link Handling Window shows links of all link types but without link attributes
- The Link Type Specific Window shows links of a specific link type with attributes belonging to it.

The following descriptions are valid for both windows if no other information is given.

NOTE

Not all menu actions are allowed on all link types. For example, the link type "HAS_SIGNAL _ELEMENT" cannot be deleted but only opened. What actions are allowed per link type are described in the reference manual "Object and Link Types".

3.5.10.1 File Menu

The File menu for the Link Handling window is shown in Figure 3-227.

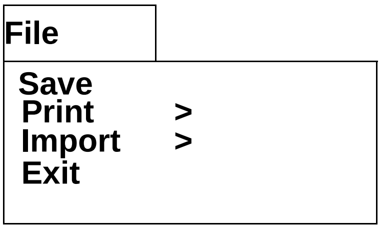


Figure 3-229. Link handler- File Menu

File - Save

Save saves entries made during the current link editing session from the work space to the database. All inputs and modifications made to the link since the last Save are stored in the database. You can also use **Accept/Save** (**<f8>**) to perform the **Save** function.

The Link Handler prompts you when the function is done:

```
1 Transaction complete -- n records posted and committed
```

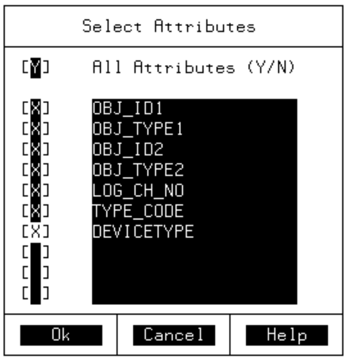
If an error occurs during the save, the whole transaction is rolled back and the cursor is positioned on the row containing the first link which causes an error.

File - Print

Print generates either a predefined (formatted) or customized (unformatted) report defined for the Link-Type (using the SQL*ReportWriter) or a printout of selected attributes. **Print** displays a secondary menu for selecting either formatted or unformatted.

File - Print - Unformatted

The **Unformatted** option writes all links displayed in the form (according to the last query condition) to a file for printing. You can select the attributes to be printed from the selection box “Select Attributes”, Figure 3-230. All attributes defined for the Link-Type (not only those actually displayed in the form) are listed in the selection box. Note that the names listed in the selection box are the display names in the form. These names are not necessarily the attribute names in the database. The output filename is requested in a dialog box, Figure 3-231. You can choose between printing out and writing only to file. Note that the ordering (of the print out or file) depends on the last execution of the Order-By selection (menu item View Order) or the default ordering after starting the Link Handler from the Structure Builder.

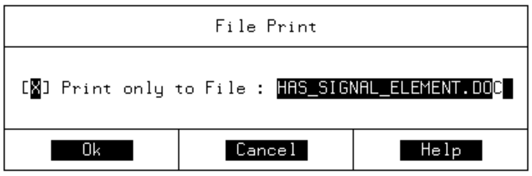


Select Attributes

[Y] All Attributes (Y/N)

[X] OBJ_ID1
[X] OBJ_TYPE1
[X] OBJ_ID2
[X] OBJ_TYPE2
[X] LOG_CH_NO
[X] TYPE_CODE
[X] DEVICETYPE
[ ]
[ ]
[ ]

Figure 3-230. Selection box “Select Attributes”



File Print

[X] Print only to File : HAS_SIGNAL_ELEMENT.DOC

Figure 3-231. Dialog box “File Print”

File - Print - Formatted

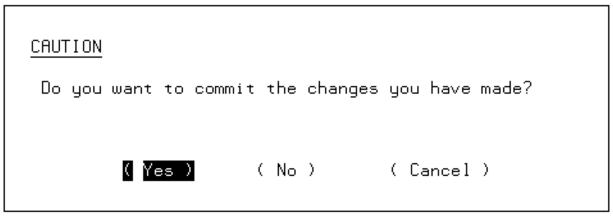
Formatted is not implemented.

File - Import - Objectlist

The **Objectlist-Into** option provides an easy way to transfer a list of objects selected within Template Builder into the Link Handler. When you invoke the corresponding menu item of the Template Builder, the object list will be written into file "objectlist.dat" located in the subdirectory ".../usr/<username>" below the actual project directory. Before you import this file into the Link Handler it can be modified with the "File Manager" utility of the Structure Builder. It is also possible to create the file "objectlist.dat" by other tools. The objects of this file can be pasted into column "OBJ_ID1" (menu item "Into Obj_id1") or column "OBJ_ID2" (menu item "Into Obj_id2") and the objects will be written into the columns starting from the current cursor position.

File - Exit

Exit closes the Link Handling window (exit the form). This command is equivalent to selecting **Exit/Cancel** (**<f7>**). If you modify data in the form and attempt to exit without saving, an alert box is displayed, Figure 3-232. This box prompts you to save the changes or cancel the operation.


CAUTION

Do you want to commit the changes you have made?

Figure 3-232. Commit alert box

Navigate to the field you want and press **Select** to activate it.**Yes** saves the pending modifications into the database. **No** discards the modification and exits the form. **Cancel** returns to where you were in the form without saving the changes.

3.5.10.2 Edit Menu

This menu provides the means to insert, copy, and delete links, and to edit data in the links attribute fields, Figure 3-233. All the links listed on the form belong to the same parent object or are part of the subtree of the parent object (With subtree = Y in Set New View... dialog box). Note that the Link Handler can not create new objects but links of existing objects.

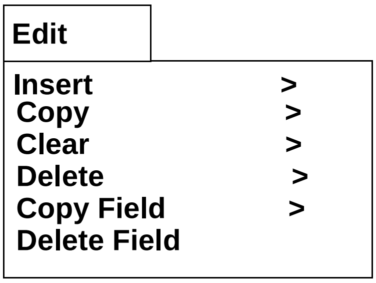


Figure 3-233. Link Handling - Edit Menu

Edit - Insert

Insert creates a new link below the current cursor position. The new link is assigned default attribute values as defined in the database (table LINK_DICT). Default values are also assigned to attributes that are not actually displayed in the form, but are defined in the table LINK_DICT. You can also use the key (<**Insert line**>) to create a new link. You must enter a unique attribute combination which identifies the new link. This attribute combination consists of OBJ_ID1, linktype e.g. IS_DOCUMENTATION_OF (only necessary for General Link Handling Window) and OBJ_ID2. You can edit the other link attributes as required.

Edit - Copy

Copy provides the means to make single or multiple copies of a selected link, or to make one copy of each link on the form starting at a selected link. **Copy** displays a menu with the following options: **Single**, **Multiple**, and **To End**.

- **Single**

  Single makes one copy of the selected link.

- **Multiple**

  Multiple makes a specified number of copies of the selected link. When you select **Multiple**, a dialog box for specifying the number of copies is displayed.

- **To End**

  To End makes one copy of all links starting at the current link, and continuing to the end of the buffer. Each copy is placed immediately after the source link.

Edit - Clear

Clear removes one or more links from the window (work space). The links still exist in the database. You can use **Clear** to unclutter the window when there are many entries, and you only want to work on a few. **Clear** does not affect the database. You cannot use **Accept/Save** to delete the links that you have cleared.

Clear displays a menu with the following options:

- **Single**

  Single removes the selected link from the display but not from the database. You can also use the key (<**Clear line**>) to clear a single link.

- **To End**

  To End removes all links from the selected link to the end of the buffer.

Edit - Delete

Delete deletes links. **Delete** displays a menu with the following options:

- **Single**

  Single deletes the selected link. The link is actually deleted from the database when you do a save (see above).

- **To End**

  To End deletes all links from the selected link to the end of the buffer. The links are actually deleted from the database when you do a save (see above).

Edit - Copy Field

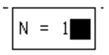
Copy Field copies the value from the field of the previous link (link above where cursor is located), and writes the value into the selected field and perhaps additional links, depending on the selected **Copy Field** option. **Copy Field** displays a menu with the following options:

- **Single**

  To copy the value to a single link, position the cursor on the line for the link that you want the value to be copied to, and on the field for the attribute to be copied. Then select **Edit-Copy Field-Single**. This copies the value from the previous link, and writes it to the corresponding field for the selected link.

- **Multiple**

  To copy the value to multiple links, position the cursor on the line for the link where you want the copy to begin, and on the field for the attribute to be copied. Then select **Edit-Copy Field-Multiple**. This displays a dialog box for specifying the number of subsequent links that you want the value copied to.



N = 1█

Figure 3-234. Dialog box "Number of Copies"

  Enter the number and then select **Accept/Save**. This copies the value from the previous link, and writes it to the corresponding field for the number of links that you specified, starting at the current link.

- **To End**

  To copy the value to all links starting at the selected link and continuing to the last link on the form, position the cursor on the line for the link where you want the copy to begin, and on the field for the attribute to be copied. Then select **Edit-Copy Field-To End**.
  This copies the value from the previous link, and writes it to the corresponding field on all links starting from the selected link and continuing to the end of the form.

Edit - Delete Field

Delete Field deletes the contents of the selected field.

3.5.10.3 View Menu

The **View** menu for Link Handling is shown in Figure 3-235.

NOTE

> The menu item **Set Undir** is only available within the **General Link Handling Window** and the menu item **Select Part** only within the **Link Type Specific Window**.

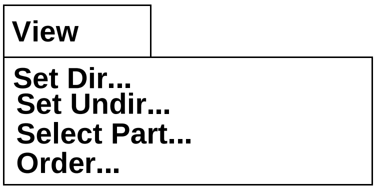


Figure 3-235. Link Handling - View Menu

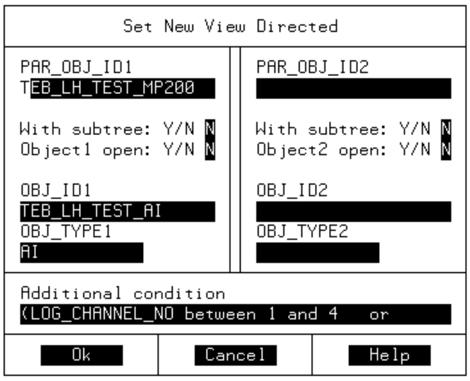
General

The Link Handler can be started from the Structure Builder in different ways. You can start the General Link Handling Window by selecting toggle button **all linktypes** or the Link Type Specific Window by selecting a specific link type within dialog box **Select Link Type** of the Structure Builder. In addition you can control the scope of link instances by selecting either a specific object in the structure, or selecting one of the menu items **Link - Of Object, - Of Next Level, - Of Whole Subtree**. Selecting of toggle button **only open links** further reduces the number of link instances.

These selected options will be written into the **Set New View Directed** or **Set New View Undirected** dialog boxes of the Link Handler and the query will be executed.

View - Set Dir

If a specific link type e.g.HAS_SIGNAL_ELEMENT is selected within the Structure Builder the values of the dialog box **Set New View Directed** Figure 3-236 are preset by the Structure Builder and the query will be automatically executed.



Set New View Directed

PAR_OBJ_ID1
TEB_LH_TEST_MP200

With subtree: Y/N N
Object1 open: Y/N N

OBJ_ID1
TEB_LH_TEST_AI
OBJ_TYPE1
AI

PAR_OBJ_ID2

With subtree: Y/N N
Object2 open: Y/N N

OBJ_ID2

OBJ_TYPE2

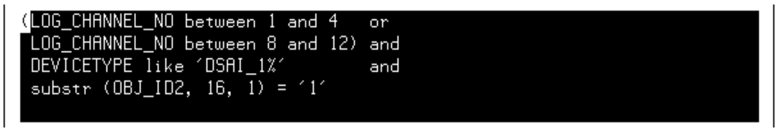
Additional condition
(LOG_CHANNEL_NO between 1 and 4   or

Figure 3-236. Dialog box "Set New View Directed"

The **Set New View Directed** dialog box can also be used to change the view on the form without having to exit the form. Calling **Set New View Directed** from the form with another object ID is equivalent to exiting the current form, selecting a new object in the Structure Builder and choosing **Link Of Object - specific link type.**

A new view may specify the right side of the link, the link type (the link type can only be specified within the General Link Handling Window), or the left side of the link. If you specify the left side (for example, **OBJ_ID1**), you retrieve only links fulfilling this condition. If you specify both sides of the link (for example, **OBJ_ID1** and **OBJ_ID2**) you retrieve only links where the object ID on the left and right sides are the same as you specified in the **Set New View Directed** dialog box. In addition to that it is possible to enter an **additional condition** (SQL Syntax) Figure 3-237.

Note that in **enter query mode** (see below) only those attributes which are not already predefined via the **Set New View Directed** dialog box are queryable.

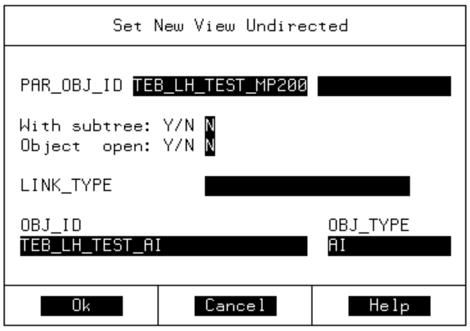


```
(LOG_CHANNEL_NO between 1 and 4   or
 LOG_CHANNEL_NO between 8 and 12) and
 DEVICETYPE like 'DSAI_1%'        and
 substr (OBJ_ID2, 16, 1) = '1'
```

Figure 3-237. Expanded field "Additional condition"

View - Set Undir

It is possible to query for links of an object independent of whether it is on the left or right side of the link. In these cases the user has to start the **General Link Handling Window** selecting toggle button **all linktypes** from the Structure Builder. If an object is selected within the Structure Builder and the Link Handler is called choosing the toggle button **all linktypes** the values of the dialog box **Set New View Undirected** Figure 3-238 are preset by the Structure Builder and the query will be automatically executed.



Set New View Undirected

PAR_OBJ_ID TEB_LH_TEST_MP200

With subtree: Y/N N
Object open: Y/N N

LINK_TYPE

OBJ_ID TEB_LH_TEST_AI
OBJ_TYPE AI

Figure 3-238. Dialog box "Set New View Undirected"

This retrieves all links of this object independent of the link type and the position of the object (left or right side) in the link.

The **Set New View Undirected** dialog box can also be used to change the view on the form without having to exit the form. Calling **Set New View Undirected** from the form with another object ID is equivalent to exiting the current form, selecting a new object in the Structure Builder and choosing **Link Of Object - all linktypes.**

Note that the **enter query mode** (see below) is not available after using **Set New View Undirected** dialog box.

View - Select Part

A Link Handling window may be divided into several parts depending how many attributes are defined for a link type. **Select Part** provides the means to select the part to view. When you choose **Select Part**, the Select Part dialog box is displayed, Figure 3-239. You can also jump into another part by navigating from field to field using the <**TAB**> key. When the last field of a certain part is entered the next <**TAB**> will jump into the first field in the next part.

Select Part is not available in the General Link Handling window since attributes are not displayed.

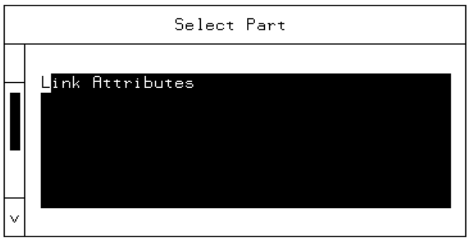
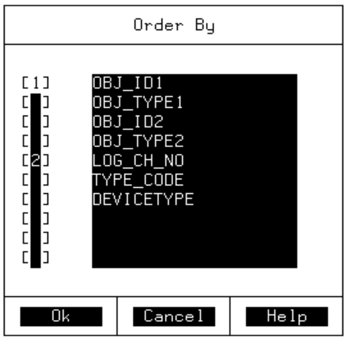


Figure 3-239. Dialog box "Select Part"

View - Order

Order provides the means to redefine the sequence of links displayed in the form. By default, links are ordered by the values of OBJ_ID1 and OBJ_ID2 in ascending order (General Link Handling Window). But some Link Type Specific Windows are ordered by other attributes. The default ordering of link type HAS_SIGNAL_ELEMENT (e.g.) is the attribute combination OBJ_ID1 and LOG_CH_NO in ascending order. **Order** displays the Order By dialog box, Figure 3-240. Use this box to assign numbers between 1 and 9 to the attributes. The links are ordered by the value of the attribute with the lowest number first. Links with the same value are ordered by the value of the attribute with the next highest number and so on. Holes (gaps) in the numbering are allowed, but a number may not be assigned to two attributes. Navigate to the attribute names and enter values between 1 and 9 into the corresponding fields of the dialog box (left of the attribute name). In Figure 3-240, the ordering sequence is: OBJ_ID1, LOG_CH_NO.



[1] OBJ_ID1
[ ] OBJ_TYPE1
[ ] OBJ_ID2
[ ] OBJ_TYPE2
[2] LOG_CH_NO
[ ] TYPE_CODE
[ ] DEVICETYPE
[ ]
[ ]
[ ]

Figure 3-240. Dialog box "Order By"

3.5.10.4 Object Menu

The Object menu for Link Handling is shown in Figure 3-241.

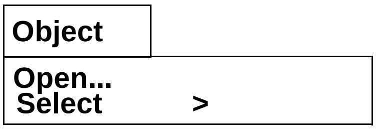


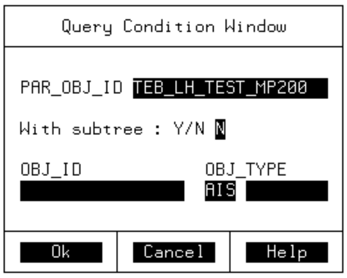
Figure 3-241. Link Handling - Object Menu

Object - Open

Open activates the Template Builder and opens the TEB view of the selected object.

Object - Select

Select provides an easy way to transfer a list of objects selected via a query within a dialog box into the Link Handler. The submenus **Into Obj_id1** or **Into Obj_id2** displays a dialog box Figure 3-242 to control the scope of objects which are copied to the selected column starting from the current cursor position**.**



PAR_OBJ_ID TEB_LH_TEST_MP200

With subtree : Y/N N

OBJ_ID

OBJ_TYPE AIS

Figure 3-242. Dialog box "Query Condition Window"

3.5.10.5 Link Menu

The Link menu for Link Handling is shown in Figure 3-243.

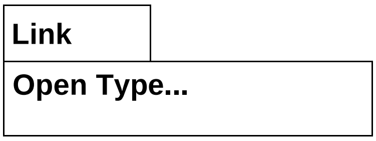
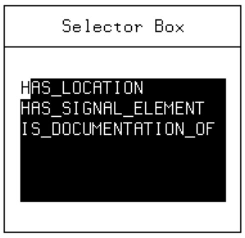


Figure 3-243. Link Handling - Link Menu

The **Link** menu is only available on the General Link Handling window. This menu is used to open a certain link type. When you select this menu, a dialog box is displayed Figure 3-244. Select the link type that you want to open. This opens a Link Type Specific window for the link type that you selected. This is equivalent to choosing **Link Of Object - specific link type** from the Structure Builder.



HAS_LOCATION
HAS_SIGNAL_ELEMENT
IS_DOCUMENTATION_OF

Figure 3-244. Dialog box "Selector Box"

3.5.10.6 Query Menu

The **Query** menu is shown in Figure 3-245.

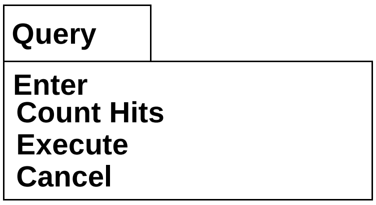


Figure 3-245. Link Handling - View Menu

Query - Enter

Enter clears the current display so you can enter query criteria directly into the form's attribute fields. Selecting **Enter** is equivalent to the **Enter Query** softkey function.

Note that in **enter query mode** only those attributes which are not already predefined via the **Set New View Directed** dialog box (see above) are queryable. If the **Set New View Undirected** dialog box (General Link Handling Window) is executed last, **enter query mode** is not available.

Query - Count Hits

Count Hits displays the number of links that a query will retrieve when executed. The count is displayed on the message line. **Count Hits** is equivalent to the **Count Hits** softkey function.

Query - Execute

Execute executes the query and retrieves the items that meet the specified query parameters. **Execute** is equivalent to the **Execute Query** softkey function.

Query - Cancel

Cancel returns to normal operation without executing the query. Selecting **Cancel** in the Enter Query mode is equivalent to the **Exit/Cancel** softkey function.

3.5.10.7 Special Key Functions

Special key functions for the link handler are described in Section 1.9.4.2, Special Key Functions.

3.5.11 Menus for the Template Builder

General

The menu of the multiple-object form is equal to the menu of the single-object form. On the precondition that the number of objects of both forms are equal (depends on the entered values of dialog boxes "Set New Parent Object" of both forms) the execution of all menu items in both forms results in the same outcome. If within both forms the same query condition is executed the number of objects stored in the forms buffer will be the same. The only difference between the multiple- and the single-object form is the number of displayed objects.

3.5.11.1 File Menu

The File menu provides commands for saving entries made to the object attribute fields from the work space of the form to the database, for generating a printout or a predefined report, exporting/importing and exiting the Template Builder, Figure 3-246.

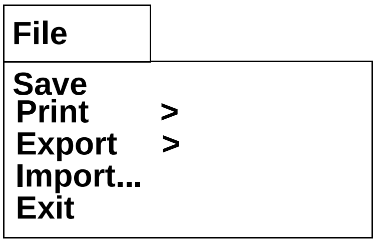


Figure 3-246. Template Builder - File Menu

File - Save

Save saves entries made during the current object editing session from the work space of the form to the database. All inputs and modifications made to the object since the last Save are stored in the database. You can also use **Accept/Save** (**<f8>**) to perform the **Save** function.

The Template Builder prompts you when the function is done:

1 Transaction complete -- n records posted and committed

If an error occurs during the save, the whole transaction is rolled back and the cursor is positioned on the row containing the first object which causes an error.

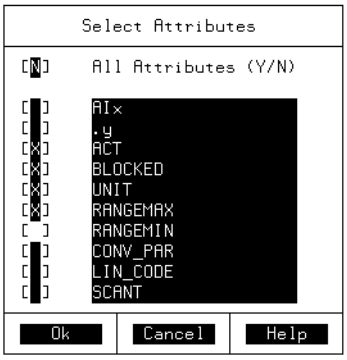
File - Print

Print generates either a predefined (formatted) or customized (unformatted) report defined for the Object-Type (using the SQL*ReportWriter) or a printout of selected attributes. **Print** displays a secondary menu for selecting either formatted or unformatted.

File - Print - Unformatted

The **Unformatted** option writes all objects displayed in the form (according to the last query condition) to a file for printing. You can select the attributes to be printed from the selection box

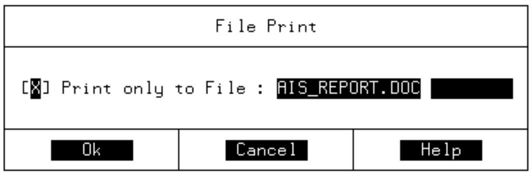
“Select Attributes”, Figure 3-247. All attributes defined for the Object-Type (not only those actually displayed in the form) are listed in the selection box. Note that the names listed in the selection box are the display names in the form.These names are not necessarily the attribute names in the database. The output filename is requested in a dialog box, Figure 3-248. You can choose between printing out and writing only to file. Note that the ordering (of the print out or file) depends on the last execution of the Order-By selection (menu item View Order) or the default ordering after starting the Template Builder from the Structure Builder.



[N] All Attributes (Y/N)

[ ] AIx
[ ] .y
[X] ACT
[X] BLOCKED
[X] UNIT
[X] RANGEMAX
[ ] RANGEMIN
[ ] CONV_PAR
[ ] LIN_CODE
[ ] SCANT

Figure 3-247. Selection box “Select Attributes”



[X] Print only to File : AIS_REPORT.DOC

Figure 3-248. Dialog box “File Print”

File - Print - Formatted

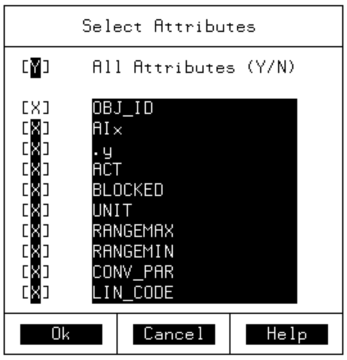
This function is not yet implemented.

File - Export

Export provides the means to export data from the AdvaBuild database into an ASCII file. The user can generate either an output file of specific attributes via attribute selection dialog or an objectlist needed for the import function of the Link Handling tool.

File - Export - User Defined

The **User Defined** option writes all objects displayed in the form (according to the last query condition) to a file. You can select the attributes to be exported from the selection box “Select Attributes”, Figure 3-249. All attributes defined for the Object-Type (not only those actually displayed in the form) are listed in the selection box. Note that the names listed in the selection box are the display names in the form.These names are not necessarily the attribute names in the database (important if you write your own select file). Note that the ordering of the file depends on the last execution of the Order-By selection (menu item View Order) or the default ordering after starting the Template Builder from the Structure Builder.



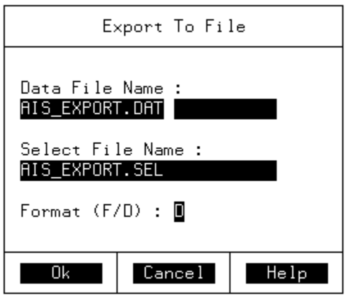
[Y] All Attributes (Y/N)

[X] OBJ_ID
[X] AIx
[X] .y
[X] ACT
[X] BLOCKED
[X] UNIT
[X] RANGEMAX
[X] RANGEMIN
[X] CONV_PAR
[X] LIN_CODE

Figure 3-249. Selection box “Select Attributes”

You must confirm your selections by pressing **OK.** This displays the Export To File dialog box, Figure 3-250. This dialog box requires the following entries:

- The Data File Name
- The Select File Name
- The Format of the data in the data file (Fixed/Delimited).



Data File Name :
AIS_EXPORT.DAT

Select File Name :
AIS_EXPORT.SEL

Format (F/D) : D

Figure 3-250. Dialog box “Export To File”

The export is started after you confirm your entries in the dialog box by selecting **OK.**
The execution consist of two steps:

- The contents of the select file will be automatically generated and written to subdirectory “.../eiu” below the actual project directory but only if the user has selected attributes via the selection box Figure 3-249
- The select file will be executed and the exported data set will be written into the specified data file located in the subdirectory “.../eiu” below the actual project directory.

If the user has not selected attributes via the selection box the contents of the select file has to be written by himself. Refer to the *Export/Import Interface* section under *Application Procedures* for a detailed description on how to use the Import/Export utility.

File - Export - Objectlist

The **Objectlist** option provides the means to transfer a list of objects into the link handling tool used for semi-automatically creation of links.

Performing this option all objects displayed in the form (according to the last query condition) will be written into file “objectlist.dat” located in the subdirectory “.../usr/<username>” below the actual project directory. Note that the ordering of the file depends on the last execution of the Order-By selection (menu item View Order) or the default ordering after starting the Template Builder from the Structure Builder. Before importing this file via the “File-Import” function of the Link Handling tool (see chapter Link Handler Menus) the user can modify this file with the “File Manager” utility of the Structure Builder.

File - Import

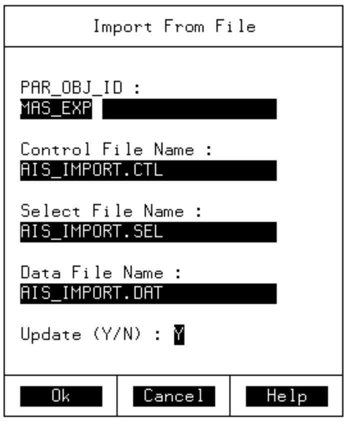
Import provides the means to import data from an ASCII file into the AdvaBuild database. Refer to the *Export/Import Interface* section under *Application Procedures* for a detailed description on how to use the Import/Export utility.

The Import From File dialog box, Figure 3-251, requires the following entries:

- The Object ID of the object which should be the parent for the imported data (PAR_OBJ_ID)
- The Control File Name
- The Select File Name
- The Data File Name
- The Update flag (Y/N).

When Update = **Y** is specified, the import updates existing objects (unique OBJ_ID) in the database.When Update = **N** is specified, all objects existing in the database are rejected.

The Import is started after you confirm your entries in the dialog box by selecting **OK**.



PAR_OBJ_ID : MAS_EXP

Control File Name : AIS_IMPORT.CTL

Select File Name : AIS_IMPORT.SEL

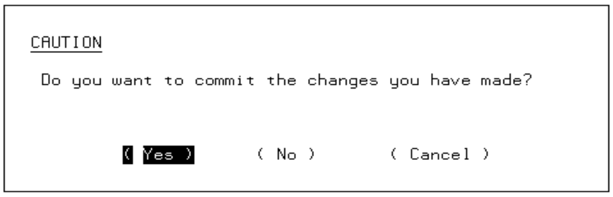
Data File Name : AIS_IMPORT.DAT

Update (Y/N) : Y

Figure 3-251. Selection box “Import From File”

File - Exit

Exit closes the Template Builder window (exit the form). This command is equivalent to selecting **Exit/Cancel** (**<f7>**). If you modify data in the form and attempt to exit without saving, an alert box is displayed, Figure 3-252. This box prompts you to save the changes or cancel the operation.



CAUTION

Do you want to commit the changes you have made?

Figure 3-252. Commit Alert Box

Navigate to the field you want and press **Select** to activate it.**Yes** saves the pending modifications into the database. **No** discards the modification and exits the form. **Cancel** returns to where you were in the form without saving the changes.

3.5.11.2 Edit Menu

This menu provides the means for inserting, copying, and deleting objects, and for editing data in the object attribute fields, Figure 3-253. All the objects listed on multiple or single object form are belonging to the same parent object or are part of the subtree of the parent object (With subtree = Y within dialog box "Set New Parent Object"). Note that the Template Builder can create only list objects but no tree objects.

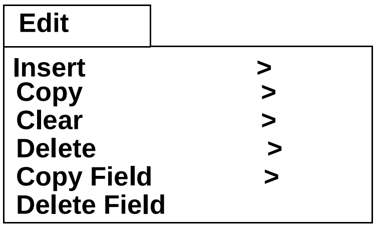


Figure 3-253. Template Builder - Edit Menu

Edit - Insert

Insert creates a new object below the current cursor position. The new object is assigned default attribute values as defined in the database (table OBJ_DICT). Default values are also assigned to attributes that are not actually displayed in the form, but are defined in the table OBJ_DICT. You can also use the key (<**Insert line**>) to create a new object. You must enter a unique OBJ_ID for the new object. You can edit the other object attributes as required. Note that the Template Builder creates the new object below the parent object given within dialog box Set New Parent Object.

NOTE

> After inserting an object, you cannot commit (save) your changes until you have assigned the new object a unique object ID.

Edit - Copy

Copy makes single or multiple copies of a selected object, or it can make one copy of each object on the form starting at a selected object. **Copy** displays a pop-up menu, Figure 3-254.

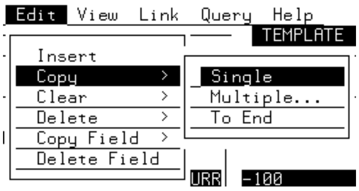


Figure 3-254. Menu item: Edit Copy

This menu provides the following copy options:

Edit - Copy - Single

To make a single copy of an object, first position the cursor on the line for the object that you want to copy, and then choose **Edit-Copy-Single**. This creates a new object below the selected object. The new object has the same object ID and attribute values as the selected object (including those attributes not actually displayed in the form). The new object is assigned to the same parent as the other objects on the form. After copying an object, you cannot commit (save) your changes until you have assigned the new object a unique object ID.

Edit - Copy - Multiple

Multiple creates *n* number of copies of the selected object following the same rules as **Edit-Copy-Single**. Select the object to be copied as described above for **Single**. Then select **Edit-Copy-Multiple**. This displays the Number of Copies dialog box, Figure 3-255.

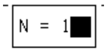


Figure 3-255. Dialog box "Number of Copies"

Enter the number of copies that you want to make, and then select **Accept/Save**. After the copies have been added to the form, you must go back and rename each one so that each new object has a unique ID.

Edit - Copy - To End

To End makes a single copy of each object, starting from the selected object and continuing to the last object on the form.

For example, consider that your form contains the following objects:

CS1_CM1
CS1_CM2
CS1_CM3.

If you position the cursor on the line for CS1_CM2 and then select **Edit-Copy-To End**, the result will be as follows:

CS1_CM1
CS1_CM2
CS1_CM2
CS1_CM3
CS1_CM3.

Edit - Clear

Clear removes one or more objects from the form (work space). The objects still exist in the database. You can use **Clear** to unclutter the form when there are many entries, and you only want to work on a few. **Clear** does not affect the database. You cannot use **Accept/Save** to delete the objects that you have cleared. **Clear** displays a pop-up menu, Figure 3-256.

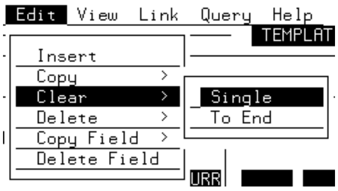


Figure 3-256. Clear Menu Options

This menu provides the following clear options:

Edit - Clear - Single

To clear a single object, position the cursor on the line for the object that you want to clear and then select **Edit-Clear-Single**. This clears the current object. You can also use the key (<**Clear line**>) to clear a single object.

Edit - Clear - to End

To clear all the objects starting at a selected object and continuing to the end of the form, position the cursor on the line that you want the clear to start at, and then select **Edit-Clear-To end**. This clears all objects from the current object up to the last object on the form.

Edit - Delete

Delete deletes objects from the database. **Delete** displays a pop-up menu, Figure 3-257.

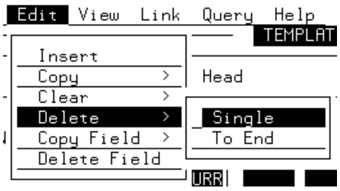


Figure 3-257. Menu item: Edit Delete

This menu has the following delete options:

Edit - Delete - Single

To delete a single object, first position the cursor on the line for the object that you want to delete, and then select **Edit-Delete-Single**. You can also use the key (<**Delete line**>) to delete a single object.

Edit - Delete - To End

To delete all objects starting from the current object, position the cursor on the line for the object where you want the delete to begin, and then select **Edit-Delete-To End**.
This deletes all objects starting from the current object to the last object on the form.

Edit - Copy Field

Copy Field copies the value for the selected attribute of the previous object (above where the cursor is positioned), and writes the value to the corresponding attribute of one or more objects below. **Copy Field** displays a pop-up menu, Figure 3-258.

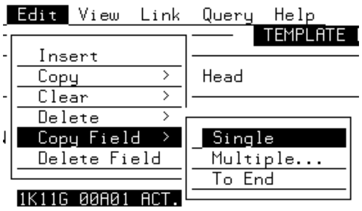


Figure 3-258. Menu item: Edit Copy Field

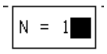
This menu provides the following copy field options:

Edit - Copy Field - Single

To copy the value to a single object, position the cursor on the attribute field for the object where you want to copy the new value to. Then select **Edit-Copy Field-Single**.
This copies the value for the selected attribute of the previous object, and writes it to the corresponding field of the selected object.

Edit - Copy Field - Multiple

To copy the value to multiple objects, position the cursor on the attribute field for the object where you want the copy to begin. Then select **Edit-Copy Field-Multiple**.
This displays a dialog box for specifying the number of objects that you want to copy the value to, Figure 3-259.



N = 1

Figure 3-259. Dialog box "Number of Copies"

Enter the number and then select **Accept/Save**. This copies the value from the previous object, and writes it to the corresponding field for the number of objects that you specified, starting at the selected object.

Edit - Copy Field - to End

To copy the value to all objects starting at the selected object and continuing to the last object on the form, position the cursor on the line for the object where you want the copy to begin, and on the field for the attribute to be copied. Then select **Edit-Copy Field-To End**. This copies the value from the previous object, and writes it to the corresponding field on all objects starting from the selected object and continuing to the end of the form.

Edit - Delete Field

Delete Field deletes the contents of the selected field.

3.5.11.3 View Menu

The **View** menu, Figure 3-260, provides the means to:

- Toggle between multiple-object form and single-object form
- Set a new parent object
- Navigate to a specific part on the template
- Define ordering criteria for the objects selected.

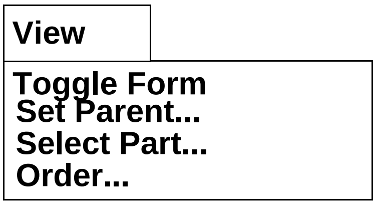


Figure 3-260. Template Builder - View menu

View - Toggle Form

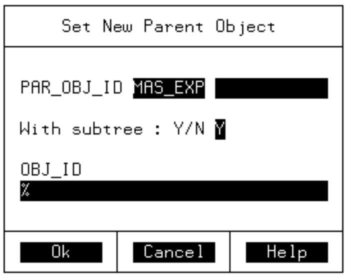
Use **Toggle form** to toggle between the multiple-object form and the single-object form. The toggle function is only allowed if all open transactions in the current form are saved into the database. That is to say, if you started editing fields on a template and did not save your changes, you will not be allowed to toggle to the other form.

Toggling from the multiple-object form to the single-object form is equal to opening the single-object form by choosing **Open** menu item from the **Object** menu in the Structure Builder. Retoggling to the multiple-object form restores the parent object id which was valid before toggling to single-object from (only important if you have changed the parent object id within single-object from).

You can also use a key function (see "Special key functions") to toggle between the multiple-object form and the single-object form.

View - Set Parent

You can use **Set parent** to change the view on the multiple- or the single-object form without having to exit the form. Calling **Set Parent** from the multiple-object form is equivalent to exiting the current form, selecting a new object as parent in the Structure Builder and choosing **Object Open Type. Set parent** provides the following dialog box, Figure 3-261.


Set New Parent Object

PAR_OBJ_ID MAS_EXP

With subtree : Y/N Y

OBJ_ID
%

Figure 3-261. Dialog Box "Set New Parent Object"

This dialog box has the following fields:

PAR_OBJ_ID	Use this field to specify the new parent object.
With subtree	Use this field to specify whether to display only the objects allocated to the next level below the new parent object (With subtree = **N**) or all objects of the same Object-Type from the whole subtree (With subtree = **Y**)
OBJ_ID	Use this field to select only a subset of these objects (pattern matching using the wildcard characters **_** and**%**).

The initial entries for this dialog box are different, depending on whether the Template Builder was started in the multiple-object form or the single-object form.

When started in the multiple-object form (Open Type):

> PAR_OBJ_ID = OBJ_ID of the object TEB has been started from, With subtree = **Y**, OBJ_ID =**%**

When started in the single-object form (Open):

> PAR_OBJ_ID = OBJ_ID of the parent object to the object TEB has been started from, With subtree = **N,** OBJ_ID = OBJ_ID of the object TEB has been started from
>
> When you use **Set parent** from the single object form, toggle to the multiple object form to see the results.

View - Select Part

Use this menu item to navigate to a selected part of the current template. When you invoke this menu item, the selection box "Select Part" is displayed, Figure 3-262. Navigate to the part you want and press [Accept/Save]. This displays the page with the first attribute belonging to the selected part.

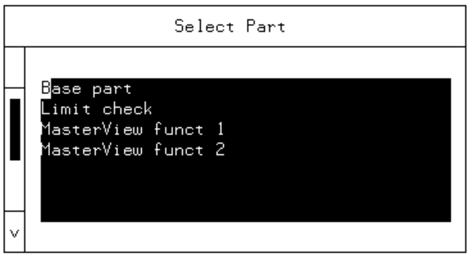
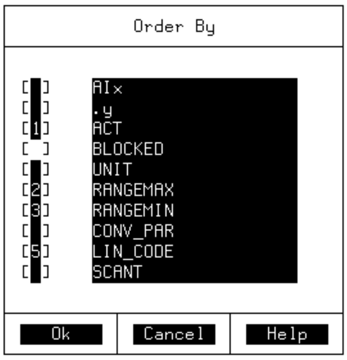


Figure 3-262. Selection box "Select Part"

You can also use a key function (see "Special key functions") to navigate to the next part or the previous part.

View - Order

Order provides the means to redefine the sequence of objects displayed in the form. By default, objects are ordered by the values of their OBJ_ID in ascending order. **Order** displays the Order By dialog box, Figure 3-263. Use this box to assign numbers between 1 and 9 to the attributes. The objects are ordered by the value of the attribute with the lowest number first. Objects with the same value are ordered by the value of the attribute with the next highest number and so on. Holes (gaps) in the numbering are allowed, but a number may not be assigned to two attributes. Navigate to the attribute names and enter values between 1 and 9 into the corresponding fields of the dialog box (left of the attribute name). In Figure 3-263, the ordering sequence is: ACT, RANGEMAX, RAMGEMIN, LIN_CODE.



[ ] AIx
[ ] .y
[1] ACT
[ ] BLOCKED
[ ] UNIT
[2] RANGEMAX
[3] RANGEMIN
[ ] CONV_PAR
[5] LIN_CODE
[ ] SCANT

Figure 3-263. Dialog Box "Order By"

3.5.11.4 Link Menu

The **Link** menu is shown in Figure 3-264.

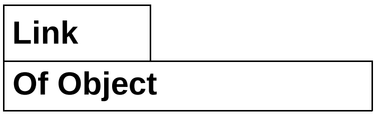


Figure 3-264. Template Builder - Link Menu

Link - Of Object

Of Object starts the Link Handling tool. Refer to Section 3.4.4, Application Procedures for Link Handling.

3.5.11.5 Query Menu

The **Query** menu is shown in Figure 3-265.

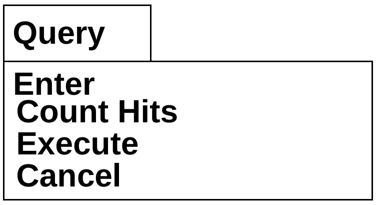


Figure 3-265. Template Builder - Query Menu

Query - Enter

Enter clears the current display so you can enter query criteria directly into the form's attribute fields. Selecting **Enter** is equivalent to the **Enter Query** softkey function.

Query - Count Hits

Count Hits displays the number of objects that a query will retrieve when executed. The count is displayed on the message line. **Count Hits** is equivalent to the **Count Hits** softkey function.

Query - Execute

Execute executes the query and retrieves the items that meet the specified query parameters. **Execute** is equivalent to the **Execute Query** softkey function. Note that only those objects would be retrieved which belong to the parent object of the dialog box "Set New Parent Object". Refer to menu "View - Set Parent" for details.

Query - Cancel

Cancel returns to normal operation without executing the query. Selecting **Cancel** in the Enter Query mode is equivalent to the **Exit/Cancel** softkey function.

3.5.11.6 Special Key Functions

Special key functions for the Template Builder are described in Section 1.9.4.2, Special Key Functions.

3.6 Data Exchange with Advant Station 100 Series Stations

3.6.1 Introduction

This section describes how to exchange data between Advant Station 500 and Advant Station 100 series Engineering Stations. Depending on your hardware and networking environment, two different configurations are possible:

- Off-line exchange between AS 500ES and AS 100ES via floppy disk
- On-line exchange via network. One AS 500ES may serve as a common File Server. Several AS 100ESs can access common project data on the AS 500ES.

Advant Station 500 and Advant Station 100 series Engineering Stations may be embedded in your network computing environment, Figure 3-266.

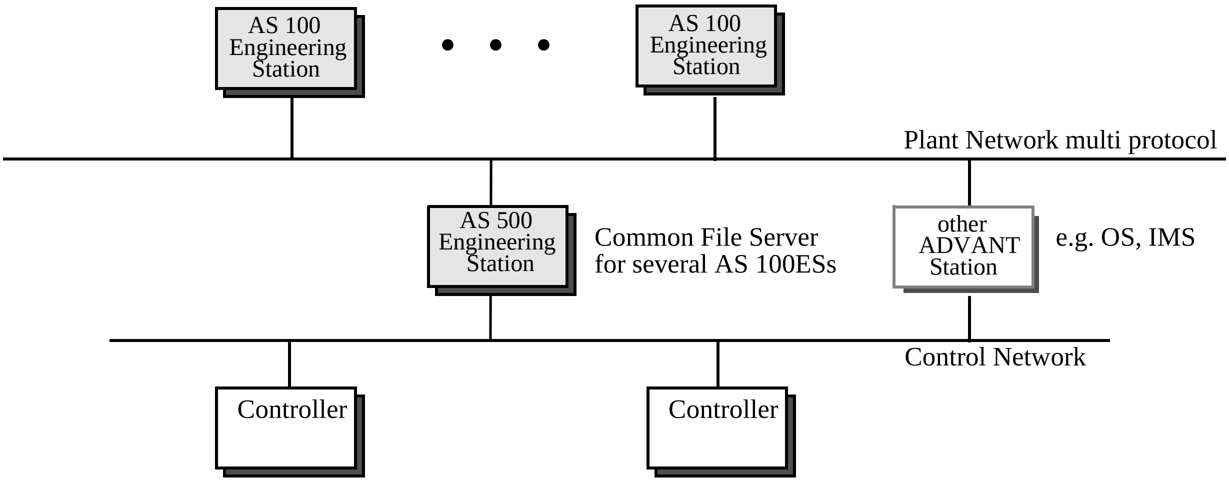

AS 100 Engineering Station
• • •
AS 100 Engineering Station
Plant Network multi protocol
AS 500 Engineering Station
Common File Server for several AS 100ESs
other ADVANT Station
e.g. OS, IMS
Control Network
Controller
Controller

Figure 3-266. Advant Station 100 and 500 series Engineering Stations embedded in a common Network Computing Environment

This figure shows a powerful configuration with one AS 500ES acting as common File Server for several AS 100ES clients.

As an alternative, data between AS 500ES and AS 100ES can be exchanged via floppy disk.

For both purposes, a special data area on the AS 500ES has been established, where AS 100ES data can be located according to AS 100ES conventions (Naming, etc.). In the following, this data area is named "AS 100ES directory structure on the AS 500ES" or simply "AS 100ES directory structure".

Figure 3-267 shows how AS 100ES accesses data of AS 500ES:

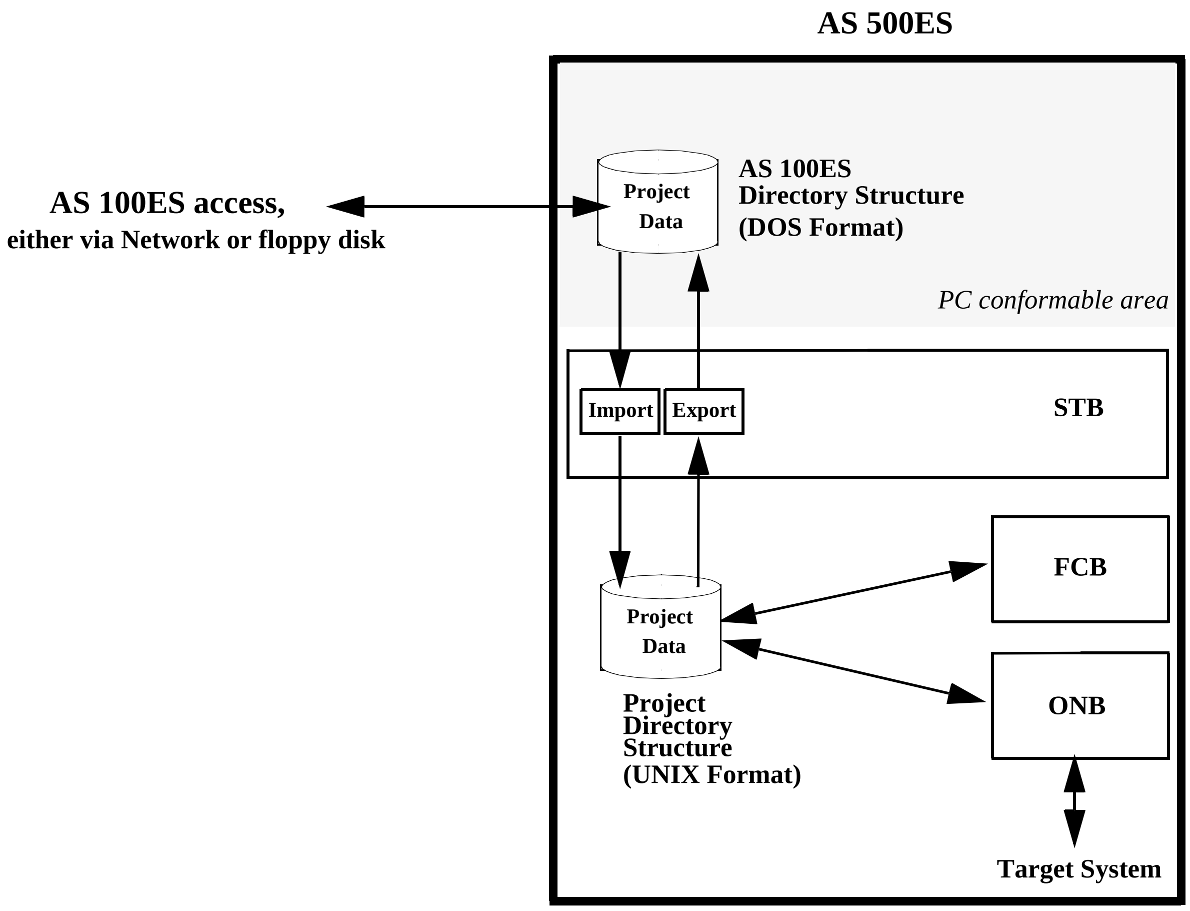


Figure 3-267. AS 100ES access to the Advant Station 500 series Engineering Station

The **AS 100ES Directory Structure** is similar to the AS 100ES directory structure and may contain the following data from AS 100ES:

1. **Libraries**
    a. General libraries (Type Circuits)

        \PC_LIB, **\TC_LEV1, \TC_LEV2**   on the PC

        These directories known from the AS100ES are equivalent to:

        /pc_lib, **/tc_lev1, /tc_lev2**   on the Workstation

    b. Project specific libraries (Type Circuits, Circuits)

        \ROOT\*PROJECTNAME***\NODES\PR_TC**
        \ROOT\*PROJECTNAME***\NODES\PR_CI**   on the PC

        These directories known from the AS100ES are equivalent to:

        /root/*projectname***/nodes/pr_tc**
        /root/*projectname***/nodes/pr_ci**   on the Workstation

2. **Project and Node specific application data**
    e.g. AMPL Source Code Files (*.AAX, *.BAX), Dump Files

        \ROOT\*PROJECTNAME***\NODENAME\***...*   on the PC

        These directories known from the AS100ES are equivalent to:

        /root/*projectname/nodename***/...**   on the Workstation

If you do not change the basic configuration, the AS 100ES directory structure on the AS500ES is located under

/users/as100es.

For a detailed description of this directory structure, see Section 3.6.6.1, AS 100ES Directory Structure on the AS 500ES. For a specification of the directories and files on the AS 100ES itself see Advant Station 100 Series Engineering Station User's Guide (document number 3BSE00508), chapter 3.

Files of the AS 100ES directory structure can be imported into and exported from the AS500ES project structure. These special **Export/Import functions for AS 100ES files** are available within the STB and include automatic UNIX to DOS conversion and vice versa, like National character conversion, File Name conversion, etc. These functions are described in Section 3.6.4, Import of AS 100ES Files and Section 3.6.5, Export of AS 100ES Files.

For detailed description of the AS 500ES Project Directory Structure, see Appendix A, AS500ES Directory Structure.

The general procedure for AS 100ES access to AS 500ES is described in Section 3.6.2, How to exchange Data via Floppy Disk and Section 3.6.3, How to establish an On-line Connection via Network.

3.6.2 How to exchange Data via Floppy Disk

If you're not using a Local Network Environment, data files can be transferred between both stations via floppy disk (see Figure 3-268).

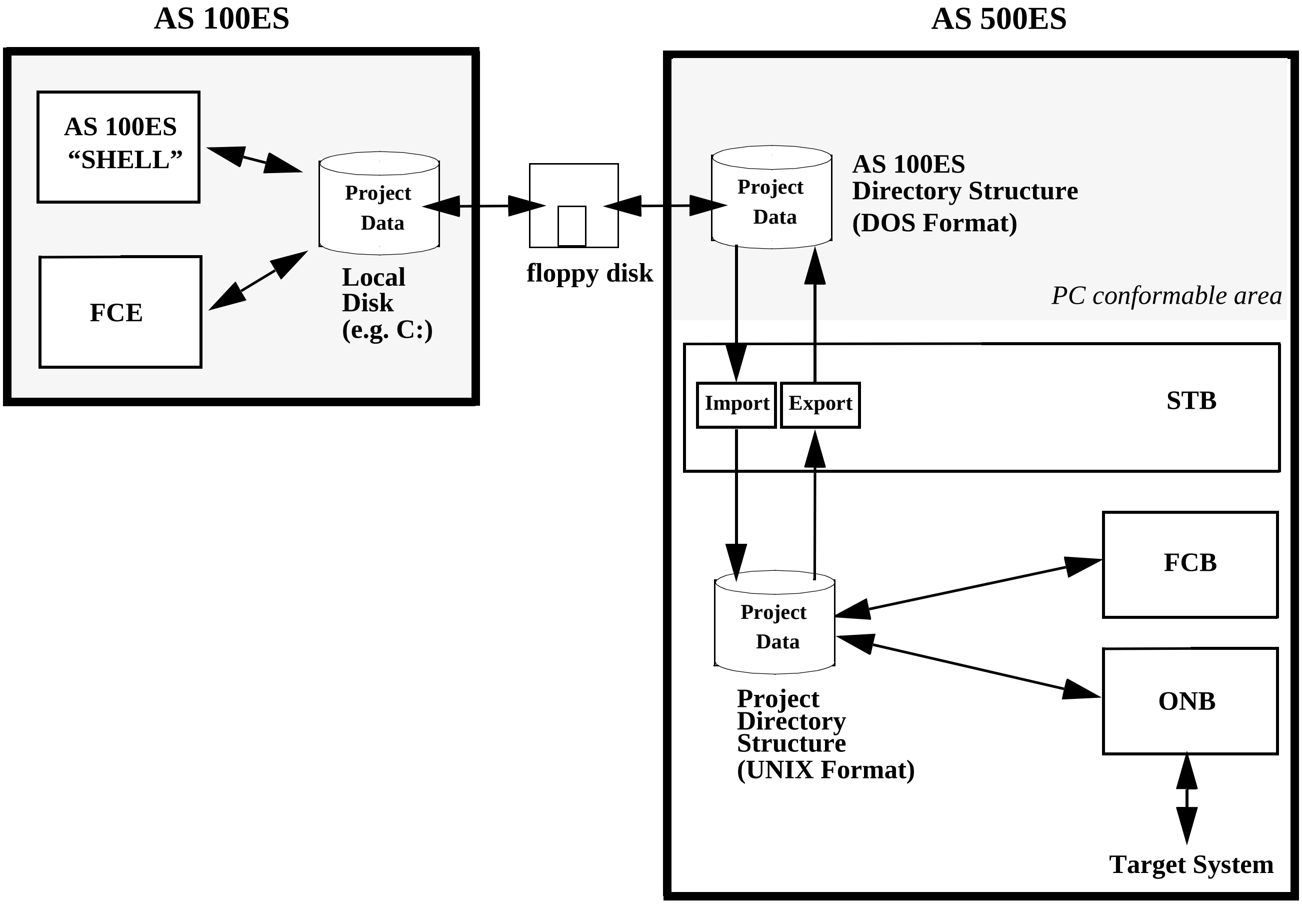


Figure 3-268. Data Exchange between AS 100ES and AS500ES via Floppy Disk

Transfer from AS 100ES to AS 500ES

To transfer from an AS 100ES to an AS 500ES:

1. At the AS 100ES, copy the selected files from the local disk to the floppy disk.
2. At the AS 500ES, copy the files from the floppy disk to the AS 100ES directory structure as follows:
   a. Change the directory to where you want the files copied to, e.g.
   `cd /users/as100es/root/`*`projectname`***`/nodes/`***`nodename`***`/pcdata`**
   b. copy the file from the floppy disk by using the **doscp** command:
   `doscp /dev/rdsk/c201d0s0:`*`file1   file1`*

NOTE

Wild cards are not allowed.

On the AS 100ES directory structure only names with lowercase letters will be recognized from the Import routine. Reason: File Name conversion, see Table 3-21 in Section 3.6.6, Details about export/import.

Transfer from AS 500ES to AS 100ES

To transfer from an AS 500ES to an AS 100ES:

1. At the AS 500ES, change the directory to where you want the files copied from, e.g.

```
cd /users/as100es/root/projectname/nodes/nodename/pcdata
```

2. Copy the file to the floppy disk by using the **doscp** command:

```
doscp file1 /dev/rdsk/c201d0s0:file1
```

3. At the AS 100ES the files must be copied from the floppy disk to the local hard disk by using DOS commands or the Windows FileManager.

NOTE

Wild cards are not allowed.

Refer to Section 3.6.6.1, AS 100ES Directory Structure on the AS 500ES for detailed description of AS 100ES Directory Structure.

3.6.3 How to establish an On-line Connection via Network

In Figure 3-269 two AS 100ESs are connected via Network to one AS 500ES. It is assumed that Network Software has been installed. This configuration has been tested using the product PC-NFS from Sun Microsystems, Inc. Also on AS 100ES side some special installation procedures have to be followed, refer to **[REFERENCE]** AS 100ES User's Guide (document number 3BSE00508), chapter 2.2.8.7 "Support for a Local Network Environment".

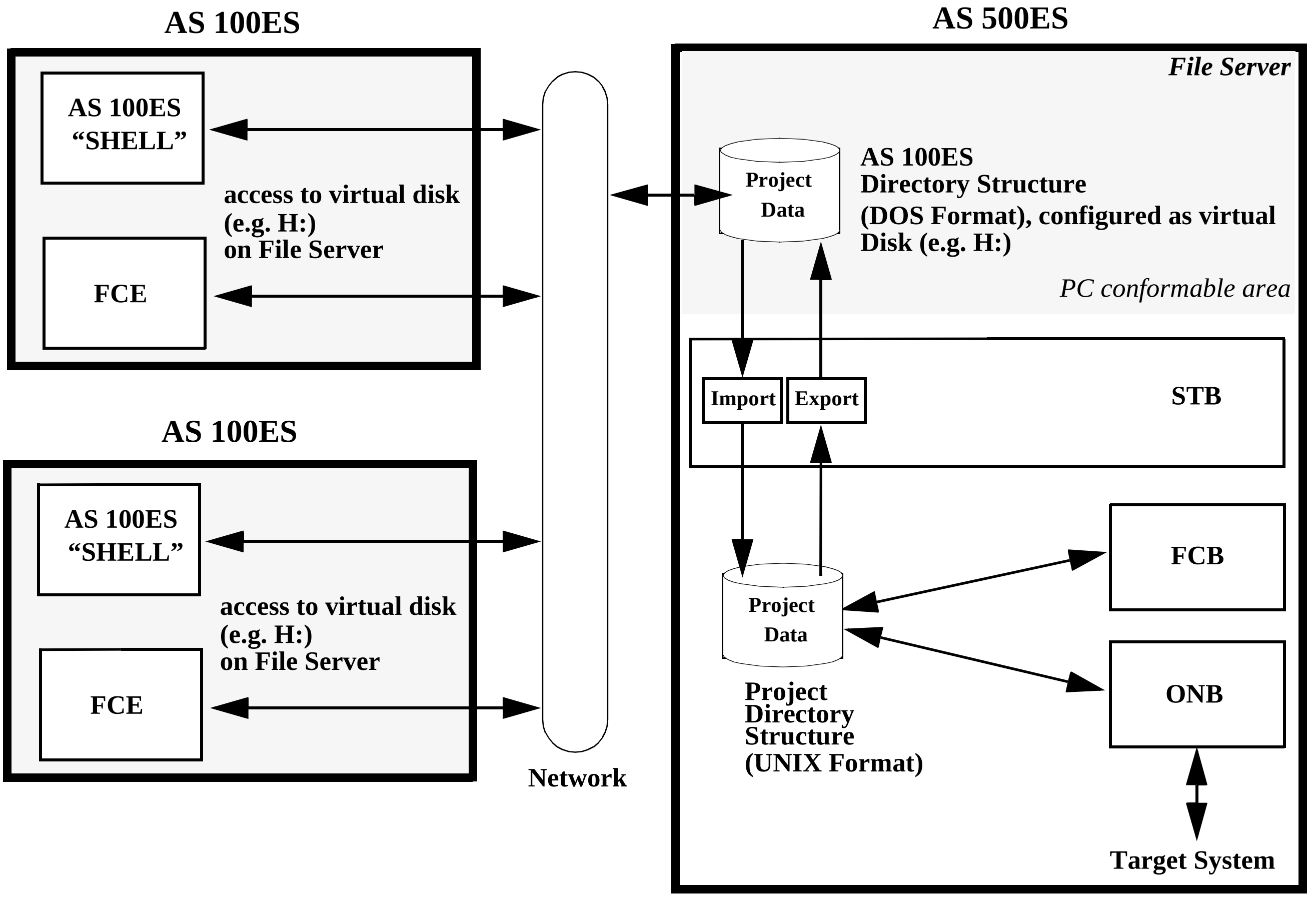

AS 100ES
AS 100ES "SHELL"
access to virtual disk (e.g. H:) on File Server
FCE
AS 100ES
AS 100ES "SHELL"
access to virtual disk (e.g. H:) on File Server
FCE
Network
AS 500ES
File Server
Project Data
AS 100ES Directory Structure (DOS Format), configured as virtual Disk (e.g. H:)
PC conformable area
Import
Export
STB
FCB
Project Data
ONB
Project Directory Structure (UNIX Format)
Target System

Figure 3-269. Data Exchange between AS 100ES and AS 500ES via Network

Using this configuration, several PC's may share the same data located on one unique AS 500ES.

3.6.4 Import of AS 100ES Files

Data Files located in the AS 100ES directory structure can be imported into the AS 500 Project Directory Structure via the Structure Builder, Figure 3-270.

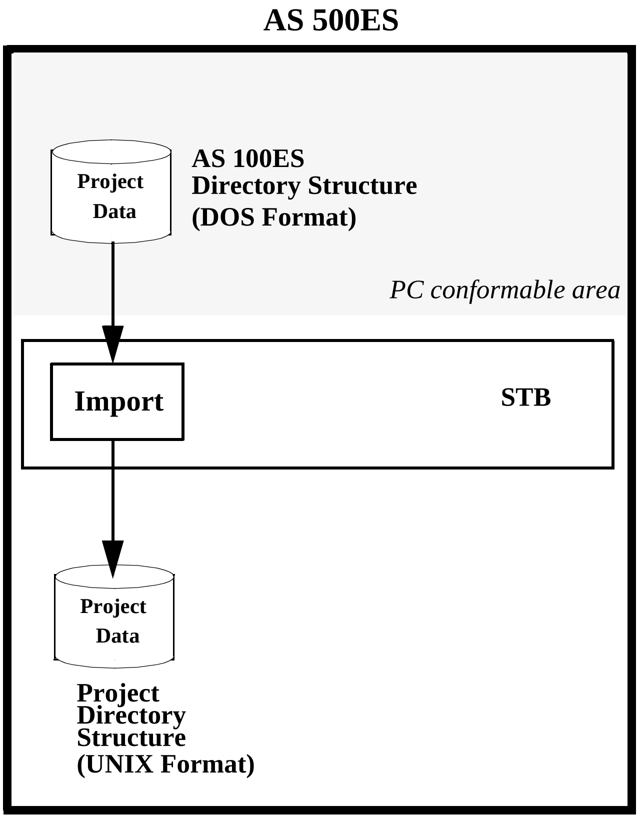

AS 500ES
Project Data
AS 100ES Directory Structure (DOS Format)
PC conformable area
Import
STB
Project Data
Project Directory Structure (UNIX Format)

Figure 3-270. Import of AS 100ES Files

The import function handles the following file extensions (in brackets: directory where they are normally located):

- *.AAX, *.TIX (PCDATA)
- *.BAX, *.TBX (DBDATA)
- *.TCS (TCDATA, TC_LEV1, TC_LEV2)
- *.CIS (CIDATA)
- *.NSS (CIDATA)
- *.PED (PC_LIB)
- *.INI of node level <NODENAME>.

To import AS 100ES files:

1. Open the Structure Builder on the project you want data to be imported into.

Import can be started with three different views:

- Import of Project Specific Data
- Import of Project Specific Libraries
- Import of Project Independent (General) Libraries.

2. If you want to import Project Independent Libraries PC_LIB, TC_LEV1, TC_LEV2: open STB on any available Master project (don't care).

Import of Node Specific Data

Dump- and source code files of a node object, Figure 3-271.

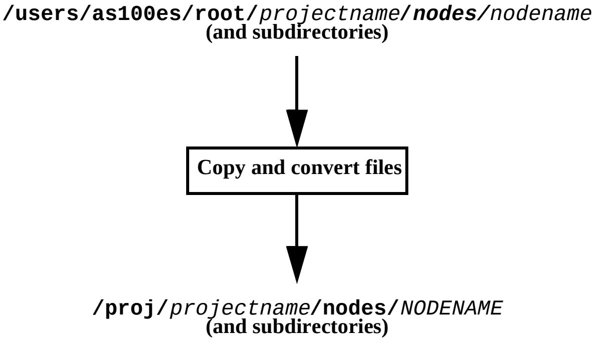

/users/as100es/root/projectname/nodes/nodename
(and subdirectories)
Copy and convert files
/proj/projectname/nodes/NODENAME
(and subdirectories)

Figure 3-271. Import Node Specific Data: affected Directories

The following node object types are supported:

- AC110C, AC410C, AC450C
- MP200, MBC200
- AS510OS, AS515OS, AS520OS
- MV810, MV830, MV850
- AS515IMS, AS520IMS
- AS515ES, AS520ES
- MG230.

For more details about node object types, refer to **[REFERENCE]** AdvaBuild Object Type Reference Manual, 3BSE001981.

It is assumed that the corresponding files are located in the AS 100ES directory structure on the AS 500ES. To import node-specific data:

1. Check that the filesystem for the node in AS 500ES project directory structure exists with all subdirectories.

   This assumes that a corresponding node object exists and the NAME attribute has been edited via TEB. If the node object does not exist, create one with STB and then edit the

NAME attribute via TEB to create the directories. The NAME attribute has to be identical with the node name on the AS 100ES.

2. Start the transfer by selecting a Node object in STB. Then choose **Special Commands** from STB's **Object** menu (see Figure 3-272).

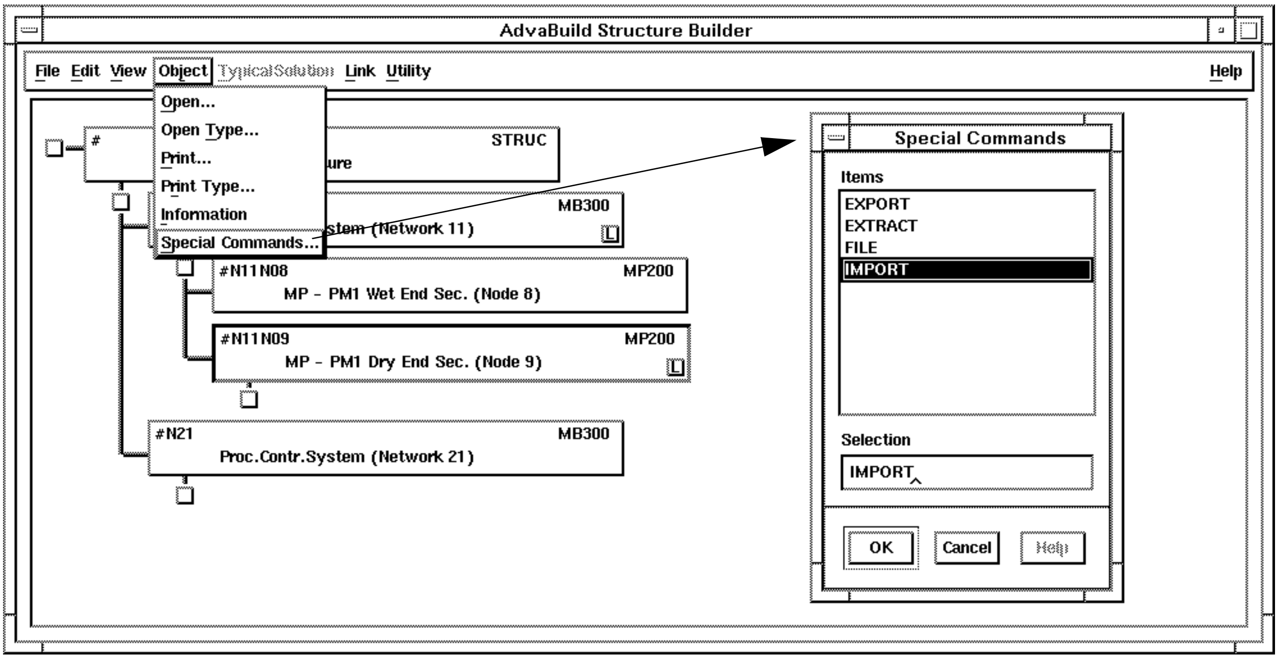
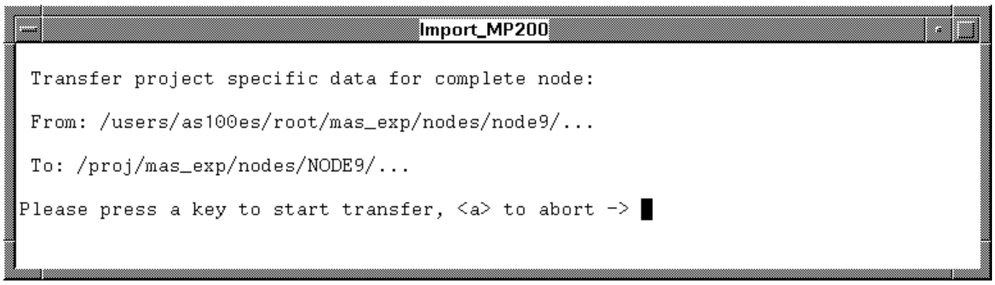


Figure 3-272. How to start Import of Node Specific Data

3. Now select item **IMPORT** from the **Special Commands** selection box and click on the **OK** button.

   This opens the import window with a prompt to start the transfer by pressing the **Return** key or to abort import by pressing the **a** key (see Figure 3-273).



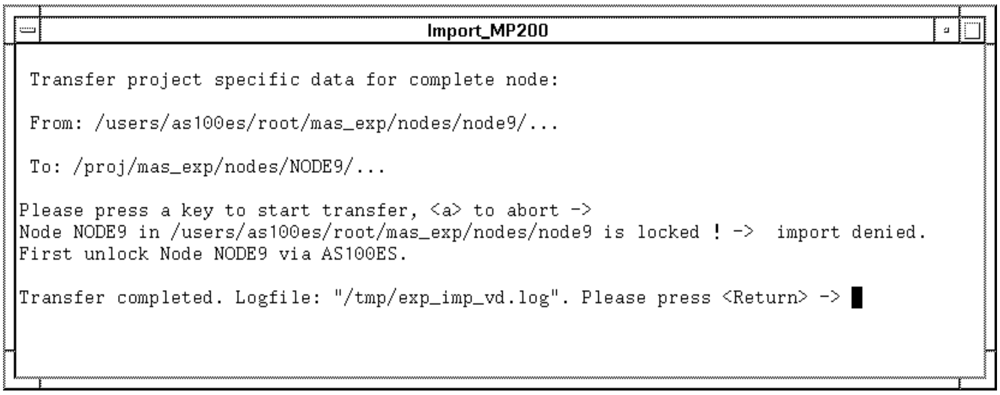
```
 Transfer project specific data for complete node:

 From: /users/as100es/root/mas_exp/nodes/node9/...

 To: /proj/mas_exp/nodes/NODE9/...

Please press a key to start transfer, <a> to abort -> █
```

Figure 3-273. Import Window

Now all datafiles found under the corresponding node directory in the AS 100ES directory structure are copied and transformed into the AS 500ES directory structure and can be used there.

NOTE

It is checked if the Node is locked by AS 100ES, import is aborted then (see Figure 3-274).



```
 Transfer project specific data for complete node:

 From: /users/as100es/root/mas_exp/nodes/node9/...

 To: /proj/mas_exp/nodes/NODE9/...

Please press a key to start transfer, <a> to abort ->
Node NODE9 in /users/as100es/root/mas_exp/nodes/node9 is locked ! ->  import denied.
First unlock Node NODE9 via AS100ES.

Transfer completed. Logfile: "/tmp/exp_imp_vd.log". Please press <Return> -> █
```

Figure 3-274. Import Window indicating that Import is denied because the Node is locked.

NOTE

It is also checked if the Node is actually in use by FCB or the Data Base Loan Flag is set by another Builder. Then a warning is issued, but user may continue import.

4. If transfer has completed successfully, start FCB or ONB, and work with the data as required.

Import of Project Specific Libraries

Project Specific data of one selected project object (PROJECT) will be copied and transformed for use on AS 500ES, Figure 3-275.

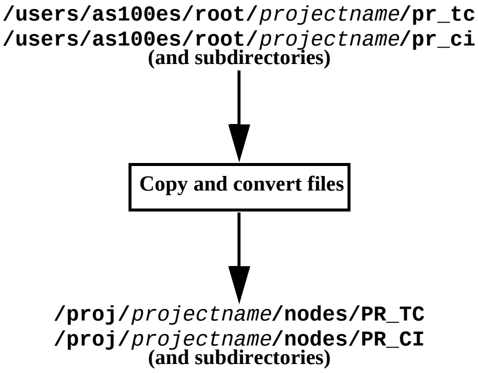

/users/as100es/root/projectname/pr_tc
/users/as100es/root/projectname/pr_ci
(and subdirectories)
Copy and convert files
/proj/projectname/nodes/PR_TC
/proj/projectname/nodes/PR_CI
(and subdirectories)

Figure 3-275. Import Project Specific Libraries: affected Directories

It is assumed that the corresponding files are located in the AS 100ES directory structure on the AS 500ES.

1. To start the transfer, select the PROJECT Object in STB. Then choose **Special Commands** from STB's **Object** menu, Figure 3-276.

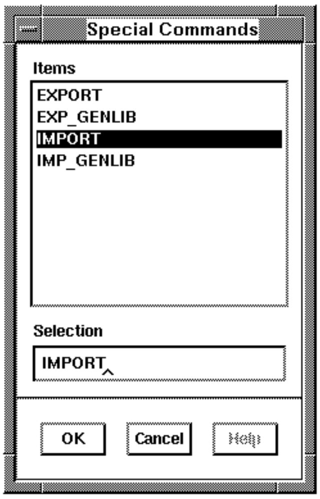


Figure 3-276. Import of Project Specific Libraries

2. Now select item **IMPORT** from the **Special Commands** selection box and click on the **OK** button. (see Figure 3-276).

   This opens the import window with a prompt to start the transfer by pressing the **Return** key or to abort import by pressing the **a** key.

   Now Project Specific Libraries are copied and transformed from the AS 100ES directory structure to AS 500ES project directory structure and can be used there.

NOTE

It is checked if a Type Circuit / Circuit is locked by AS 100ES, import is aborted then.

3. If import completed successfully, start FCB or ONB, and work with the new Type Circuits / Circuits of Project Specific Libraries as required.

Import of General Libraries

Project Independent Libraries will be copied and transformed from AS 100ES for use on AS 500ES, Figure 3-277.

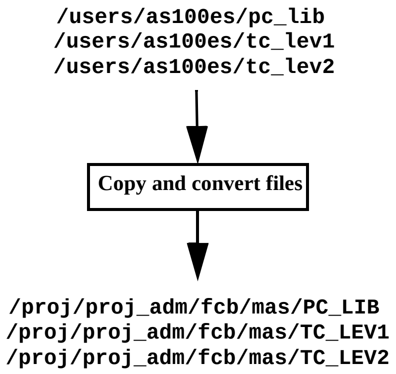

/users/as100es/pc_lib
/users/as100es/tc_lev1
/users/as100es/tc_lev2
Copy and convert files
/proj/proj_adm/fcb/mas/PC_LIB
/proj/proj_adm/fcb/mas/TC_LEV1
/proj/proj_adm/fcb/mas/TC_LEV2

Figure 3-277. Import General Libraries: affected Directories

It is assumed s that the corresponding files are located in the AS 100ES directory structure on the AS 500ES.

1. To start the transfer, select the PROJECT Object in STB. Then choose **Special Commands** from STB's **Object** menu.
2. Now select item **IMP_GENLIB** from the appearing **Special Commands** selection box and click on the **OK** button.

   This opens the import window with a prompt to start the transfer by pressing the **Return** key or to abort import by pressing the **a** key.

   Now Project Independent Libraries are copied and transformed from the AS 100ES directory structure to AS 500ES project directory structure and can be used there.

   NOTE

   It is checked if a Type Circuit / Circuit is locked by AS 100ES, import is aborted then.

3. If Import completed successfully, start FCB or ONB, and work with the new Type Circuits of Project Independent Libraries as required.

3.6.5 Export of AS 100ES Files

AS 100ES relevant Data Files located in the AS 500ES project directory structures can be exported to the AS 100ES directory structure via the Structure Builder, Figure 3-278.

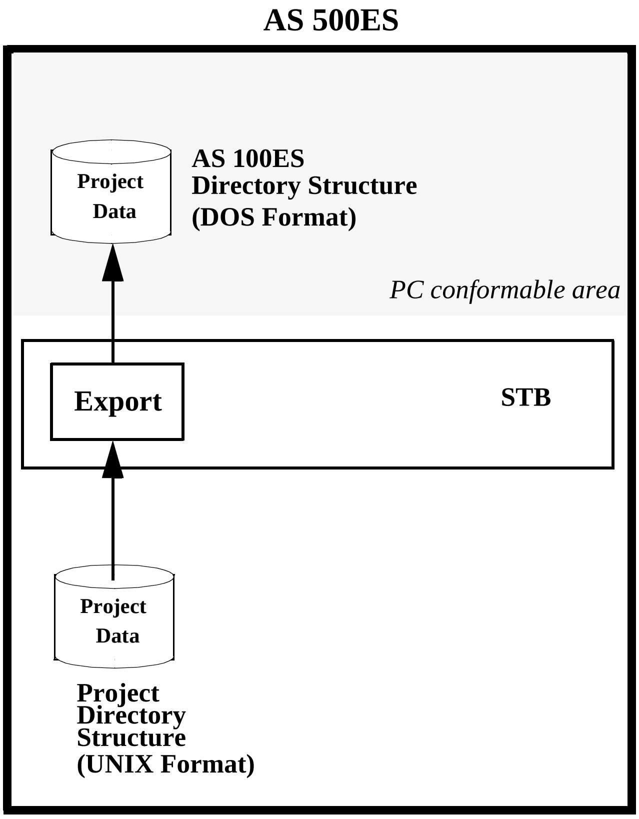


Figure 3-278. Export of AS 100ES Files

The export function handles the following file extensions (in brackets: directory in the AS 500ES project directory structure where they are normally located):

- *.AAX, *.TIX (PCDATA)
- *.BAX, *.TBX (DBDATA)
- *.TCS (TCDATA, TC_LEV1, TC_LEV2)
- *.CIS (CIDATA)
- *.NSS (CIDATA)
- *.PED (PC_LIB)
- *.INI of node level <NODENAME>.

1. Open the Structure Builder on the project you want data to be exported from.

   Export can be started with three different views:

- Export of Project Specific Data
- Export of Project Specific Libraries
- Export of Project Independent Libraries.

If you want to export general libraries PC_LIB, TC_LEV1, TC_LEV2: open STB on any available Master project (don't care).

Export of Node Specific Data

Dump- and source code files of a node object, Figure 3-279.

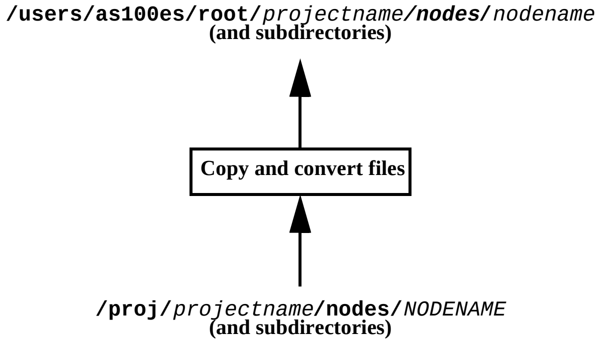

/users/as100es/root/projectname/nodes/nodename
(and subdirectories)
Copy and convert files
/proj/projectname/nodes/NODENAME
(and subdirectories)

Figure 3-279. Export Node Specific Data: affected Directories

The following node object types are supported:

- AC110C, AC410C, AC450C
- MP200, MBC200
- AS510OS, AS515OS, AS520OS
- MV810, MV830, MV850
- AS515IMS, AS520IMS
- AS515ES, AS520ES
- MG230.

For more details about node object types, refer to AdvaBuild Object Type Reference Manual.

1. Start the transfer by selecting a Node object in STB. Then choose **Special Commands** from STB's **Object** menu
2. Now select item **EXPORT** from the **Special Commands** selection box and click on the **OK** button.

   This opens the export window with a prompt to start the transfer by pressing the **Return** key or to abort export by pressing the **a** key.

   Now node data and subdirectories are copied and transformed into the AS 100ES directory structure.

NOTE

It is checked if the node is locked by AS 100ES, export is aborted then.

3. If export successfully completed, you can access these files from AS 100ES either directly or via floppy disk transfer, as described in Section 3.6.2, How to exchange Data via Floppy Disk.

   Now you can work with the data from the AS 500ES in AS 100ES by using the tools FCE and COF as required.

Export of Project Specific Libraries

Project Specific data of one selected project object (PROJECT) will be copied and transformed for use on AS 100ES, Figure 3-280.

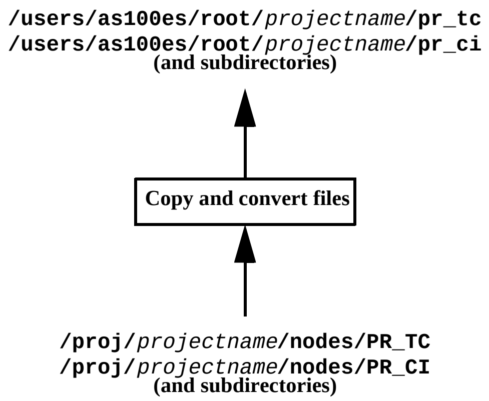

/users/as100es/root/projectname/pr_tc
/users/as100es/root/projectname/pr_ci
(and subdirectories)
Copy and convert files
/proj/projectname/nodes/PR_TC
/proj/projectname/nodes/PR_CI
(and subdirectories)

Figure 3-280. Export Project Specific Libraries: affected Directories

1. To start the transfer, select the PROJECT object in STB. Then choose **Special Commands** from STB's **Object** menu.
2. Now select item **EXPORT** from the appearing **Special Commands** selection box and click on the **OK** button.

   This opens the export window with a prompt to start the transfer by pressing the **Return** key or to abort export by pressing the **a** key.

   Now project specific data and subdirectories are copied and transformed into the AS 100ES directory structure.
3. If export successfully completed, you can access these files from AS 100ES either directly or via floppy disk transfer, as described in Section 3.6.2, How to exchange Data via Floppy Disk.
4. Change to work on AS 100ES. If you have no corresponding Project or Node on AS 100ES, create one.

   Now you can work with the data from the AS 500ES in AS 100ES by using the tools FCE and COF as required.

Export of General Libraries

Project Independent Libraries will be copied and transformed from AS 500ES for use on AS 100ES, Figure 3-281.

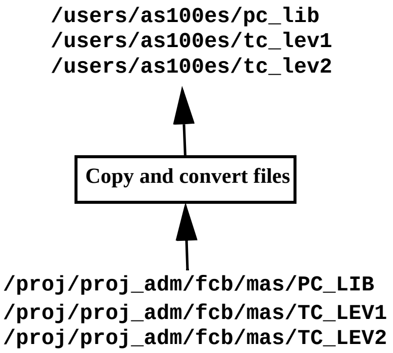

/users/as100es/pc_lib
/users/as100es/tc_lev1
/users/as100es/tc_lev2
Copy and convert files
/proj/proj_adm/fcb/mas/PC_LIB
/proj/proj_adm/fcb/mas/TC_LEV1
/proj/proj_adm/fcb/mas/TC_LEV2

Figure 3-281. Export General Libraries: affected Directories

1. To start the transfer, select the PROJECT object in STB. Then choose **Special Commands** from STB's **Object** menu.
2. Now select item **EXP_GENLIB** from the appearing **Special Commands** selection box and click on the **OK** button.

   This opens the export window with a prompt to start the transfer by pressing the **Return** key or to abort export by pressing the **a** key.

   Now Project Independent Libraries are copied and transformed into the AS 100ES directory structure.
3. If export successfully completed, you can access these files from AS 100ES either directly or via floppy disk transfer, as described in Section 3.6.2, How to exchange Data via Floppy Disk.

   Now you can work with the data from the AS 500ES in AS 100ES by using the tools FCE and COF as required.

3.6.6 Details about export/import

3.6.6.1 AS 100ES Directory Structure on the AS 500ES

The location of the AS 100ES directory structure on the AS 500ES is defined by a UNIX environment variable (**C_AS100ES**). This environment variable is set during UNIX login. If you do not change the standard configuration, this environment variable is set to **`/users/as100es.`**

To check that, enter in the hpterm which is displayed after login:

```
echo $C_AS100ES
```

In this document it is assumed that this environment variable points to

```
/users/as100es.
```

If you want to use a AS 500ES file server for your AS 100ES environment check the default value and use directory `/users/as100es/root`

This directory and the corresponding subdirectories may not exist when starting export/import. In this case, export/import behaves as follows:

- if you want to import data, the selected function cannot be performed since the corresponding source directories do not exist.
- if you want to export data, the corresponding directory structure will be automatically created.

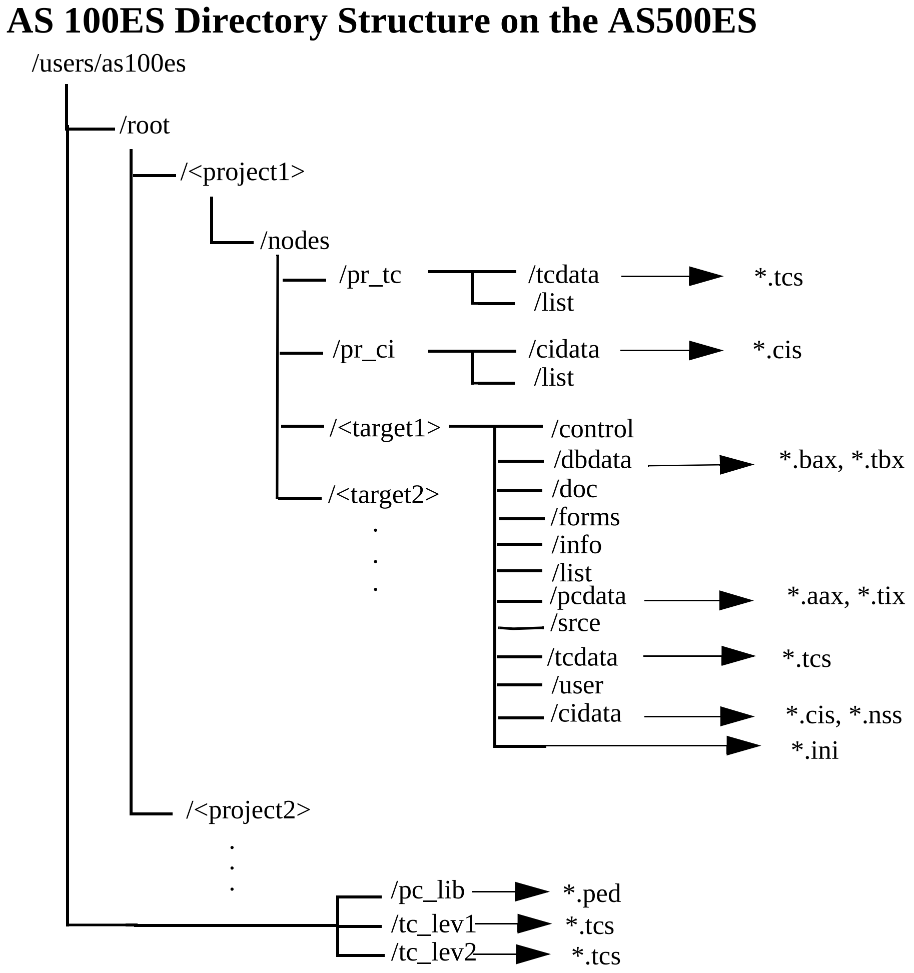


Figure 3-282. AS 100ES Directory Structure on the AS 500ES

For a specification on the directories and files on the AS 100ES itself see **[REFERENCE]** AS 100ES User's Guide (document number 3BSE00508), chapter 3.

3.6.6.2 Automatic Conversion during Export/Import

Import / export include automatic conversion routines of all transferred files. This conversion includes:

- General DOS to UNIX and UNIX to DOS conversion of file contents:

  The UNIX utilities ux2dos and dos2ux are used here.

- Conversion of file name:

Table 3-21. File Name Conversion

AS 100ES*	**AS 500ES**
PC file name	**Workstation file name**
small letters*	capital letters

NOTE

*"AS 100ES" is the AS 100ES directory structure on the AS 500ES. According to PC-NFS conventions, those files have to be written in small letters.
On AS 100ES itself, file names are all displayed in capital letters (DOS doesn't separate between small and capital letters).

- Filter for national character set:

  In AS 500ES Release 2.6 supported languages are Swedish, US-English and German.
  If source code files shall be handled in the correct way, the National Language Settings on the AS 100ES and on the AS 500ES must be known.

  Language settings on AS 100ES and AS 500ES should be the same to secure a correct interpretation of national characters during export / import.

  – On the AS 100ES, refer to the AS 100ES manual.

  – The switch for National Language Setting on the AS 500ES is evaluated in the following way:

    If in the current directory on the AS 500ES a file named **c_language.dat** exists, the content of this file determines the language (normally for export / import of Node Specific Data, in the node directory). If no such file exists in the current directory, the UNIX environment variable **C_LANGUAGE** is taken as language setting.

To import/export source code between AS 100ES and AS 500ES with different national code settings, the following assignment must be considered:

Table 3-22. Conversion of National Characters

AS 100ES / AS 500ES					
Sweden	**HP**	**German**	**HP**	**US-English**	**HP**
Ä	C4	Ä	C4	[	5B
Ö	D6	Ö	D6	\	5C
Å	C5	Ü	DD	}	5D
ä	E4	ä	E4	{	7B
ö	E6	ö	E6	|	7C
å	E5	ü	EC	}	7D
		ß	DF	~	7E

3.6.6.3 Constraints and Dependencies

Naming Conventions for Files in the AS 100ES Directory Structure

- From workstation point of view (HP-UX):
  - Only 8 lower case characters with 3 lower case extension characters have to be used
- From PC point of view (DOS):
  - Only upper case characters have to be used.
  - If AS 100ES is connected to the AS 500ES via network:
    All lower case characters in the HP-UX system´s virtual disk are automatically seen as upper case characters in the DOS system (PC-NFS convention).

Assumed Software Configurations

- AS 500ES:
  AS 500ES Software Release 2.6
- AS 100ES
  AdvaBuild for Windows Release 1.2.

3.7 On-line Builder

3.7.1 Introduction

This section describes the On-line Builder of the Advant Station 500 Engineering Station.

The On-line Builder may be used to load application programs via the MasterBus 300 into MasterPiece 200, MasterView 800, MasterGate 230, AdvantController 400C, AdvantStation 500ES, AdvantStation 500OS, AdvantStation 500IMS and MasterBatchController 200 stations. Those programs may be tested and debugged with the On-line Builder.

The On-line Builder is not designed to create and edit application programs. This work has to be done with engineering tools like Structure Builder and Function Chart Builder.

The On-line Builder is not designed to load and debug application programs for AdvantStation 110C stations. This functionality is part of the Function Chart Builder for AdvantStation 110C.

The description of the commands for loading and debugging application programs is not part of this section. The description of these Commands can be found in the ABB Master documents (see Section 1.5, Related Documentation).

To work with the On-line Builder, you must have some knowledge of the Structure Builder and configuration and programming of MasterPiece 200 and the other application systems for On-line Builder.

Equipment Requirements

The On-line Builder software, together with the RealTime Accelerator board hardware, is an optional package to the Advant Station 500 Engineering Station.

To run the On-line Builder software, the RealTime Accelerator board has to be installed in the Engineering Station, connecting it to the MasterBus 300.

3.7.2 Preparation

During installation of the Advant Station 500ES the installed RealTimeAccelerator board must be configured. The RealTime Accelerator board must be assigned a Net and Node number and a printer must be given. After installation the board must be booted (see below).

3.7.3 How to Boot the RealTimeAccelerator

The RealTime Accelerator board must always be booted after a machine boot. After HP login the boot function checks if the RealTime Accelerator board was booted before. If not a window appears:

"Do you wish to activate the RealTimeAccelerator board for MASTER (y/n) ?"

Press <**y**> to set up the RealTime Accelerator board. After the boot process an OK or error message appears. Continue the session with <**Return**>.

If an error was detected then read the logfile in /tmp. Then logout and try it again.

If the AdvaBuild On-line Builder with RealTime Accelerator board is in a undefined condition, then boot the board:

1. In the hp-terminal, type **$C_BIN_ONF_ETC/rta_boot <Return>**.
2. Answer the question by typing **y** (without <**Return**>).
3. If you get the message "Board 1 still locked, continue (y/n)?', then type **y <Return>**.
4. If the next message reads 'Please press <Return> to continue', then press the **Return** key. The problem should be solved now.
5. If you get an error message, then read the log-file in order to find out what happened. The log-file has the name RTABootlog.extension. The extension is a combination of the date and the time when the file has been created (e.g. 940412-8:15). The log-file is stored in the directory of /tmp.

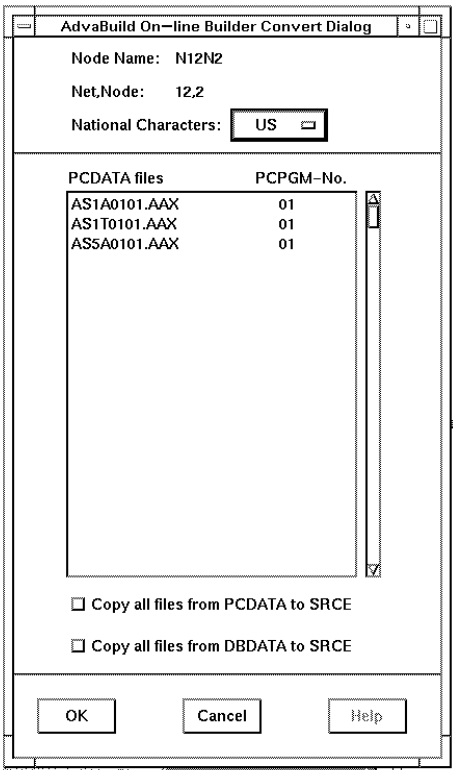
3.7.4 How to Start the On-line Builder

The On-line Builder is started by opening the ON-LINE view of a target object in the Structure Builder. The target objects are MP200, PCPGM, MV8**, MG230, AC4**C, AS 500ES, AS 500OS, AS 500IMS and MBC200. To start the On-line Builder:

In the Structure Builder tree presentation double click on the target object you want to work with. This displays the Open Object dialog box. Click on the **ON-LINE** entry in the **Views** list. Then click on the **OK** button.

3.7.5 On-line Builder Convert Dialog window

The AdvaBuild On-line Builder Convert Dialog window (Figure 3-283) shows the Node Name and the Net, Node number of the selected target object. The National Characters menu shows as default selection the language as defined in the file **c_language.dat** of the object. The list in the window shows the PCDATA files (only *01.AAX files). In the column on the right of the list, you will see the number of the associated PCPGM object. If you see an asterisk (*) instead of a valid number, then you have a PC source code file which is not associated to an existing PCPGM object.



AdvaBuild On–line Builder Convert Dialog

Node Name: N12N2

Net,Node: 12,2

National Characters: US

PCDATA files	PCPGM-No.
AS1A0101.AAX	01
AS1T0101.AAX	01
AS5A0101.AAX	01

☐ Copy all files from PCDATA to SRCE

☐ Copy all files from DBDATA to SRCE

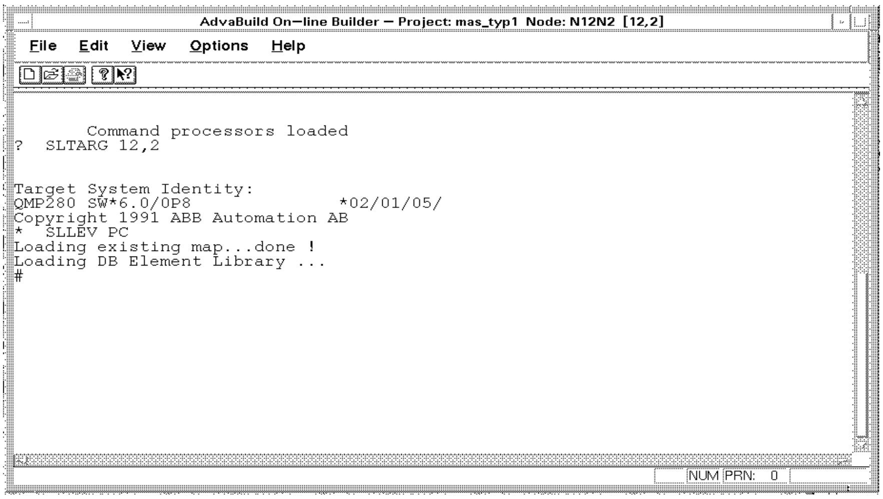
Figure 3-283. AdvaBuild On-line Builder Convert Dialog

Selecting List Objects

To select just one list object, click the left mouse button on a row in the list window. The selected row is highlighted to indicate selection. To deselect an object, click control left mouse button. To select more than one list object, first hold the left mouse button down and move the mouse cursor to the bottom of the list. When you release the mouse button all highlighted list objects are selected. You can also select multiple objects by selecting one object, holding down <**Shift**>, and then selecting another object. This operation selects all objects between the two selected objects.

Items can be added to or deleted from a selected range by using <**Ctrl**>. To add an additional range to an existing selection, move to the first item of the new group, press and hold <**Ctrl**>, and then press the left mouse button. The item under the pointer inverts; any previous selections are unaffected. This item becomes the initial item for the new selection range. If the pointer is dragged through additional items while <**Ctrl**> and left mouse button are held down, those items invert as described above.

The check button "Copy all files from PCDATA to SRCE" can be selected to copy all AAX, TIX and TCS files of the node. The second check button "Copy all files from DBDATA to SRCE" with the BAX and TBX files performs the same function.


AdvaBuild On—line Builder — Project: mas_typ1 Node: N12N2 [12,2]
File
Edit
View
Options
Help
NUM
PRN: 0

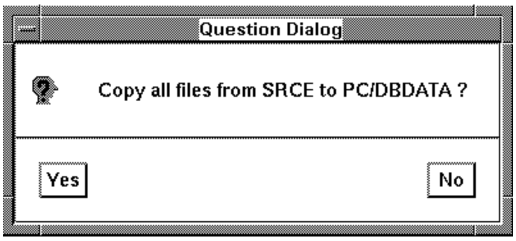
```
      Command processors loaded
?  SLTARG 12,2


Target System Identity:
QMP280 SW*6.0/0P8              *02/01/05/
Copyright 1991 ABB Automation AB
*  SLLEV PC
Loading existing map...done !
Loading DB Element Library ...
#
```

Figure 3-284. On-line Builder window

Prepare Source Code for Backtranslate

If you up-loaded source code files from the target node which shall be integrated later into the project structure (via Backtranslate), then these files first have to be copied from the directory of SRCE and USER to the directors of DBDATA, PCDATA and TCDATA.

After having disconnected from the target node, you will see a dialog box as in Figure 3-285. There is no selection of files, you may only copy all files or no file at all.



Copy all files from SRCE to PC/DBDATA ?

Figure 3-285. Convert source code selection

Click on the **Yes** button if you want all files to be copied. Click on the **No** button if you don't want any file to be copied.

3.7.6 How to Prepare Source Code for On-line Builder

If you want to down-load source code into the target node (which you edited with the Function Chart Builder and prepared with Generate Source for example), then the source code files first have to be copied from the directors of DBDATA, PCDATA and TCDATA to the directory SRCE and USER. In the AdvaBuild On-line Builder Convert Dialog window (Figure 3-283), you can select which files you want to copy. This copy transaction includes character converting from 8 bit to 7 bit, compressing of TIX and TBX files and splitting of big source code files (>400 kbyte). Little source code files (<400 kbyte) will be merged to any with about 400 kbyte.

Language selection

Source code files in the directors of DBDATA and of PCDATA are coded in the 8 bit ISO88591 code, whereas source code files in the directory of SRCE are coded in the 7 bit ASCII code. Source code files may contain special national characters like Å and Ü. When converting and copying source code files, the Convert Dialog must know the selected language in order to handle those national characters correctly.

Language US:	
8 bit code	**7 bit code**
80 - FF	2E = .

Language Swedish:	
8 bit code	**7 bit code**
C4 = Ä	5B = [
D6 = Ö	5C = \
C5 = Å	5D =]
E4 = ä	7B = {
F6 = ö	7C = |
E5 = å	7D = }
all other >= 80	2E = .

Language German:	
8 bit code	**7 bit code**
C4 = Ä	5B = [
D6 = Ö	5C = \

Language German:	
DC= Ü	5D =]
E4 = ä	7B = {
F6 = ö	7C = |
FC = ü	7D = }
DF = ß	7E = ~
all other >= 80	2E = .

As a default selection, the Convert Dialog will use the language as defined in the file **c_language.dat** of the object. But the user is allowed to make a different language selection. This language selection can only be done when entering the On-line Builder. It applies both for copy from DBDATA, PCDATA and TCDATA to SRCE and USER and for copy from SRCE and USER to DBDATA, PCDATA and TCDATA. The Convert Dialog will change the file **c_language.dat** according to your selection.

- In the Convert Dialog (Figure 3-283) you will see the text 'National Characters:'. Click on the button to the right of this text. You will see a choice of different languages.
- Click on the language you want to select. The selected language will be display on the button.

3.7.7 How to Work with the On-line Builder

Descriptions of the Commands can be found in the ABB Master documents.

This section provides a description of some special rules which apply when calling Commands on the Advant Station 500 Engineering Station.

SLTARG

You can use this Command to change from one target node to another.

CAUTION

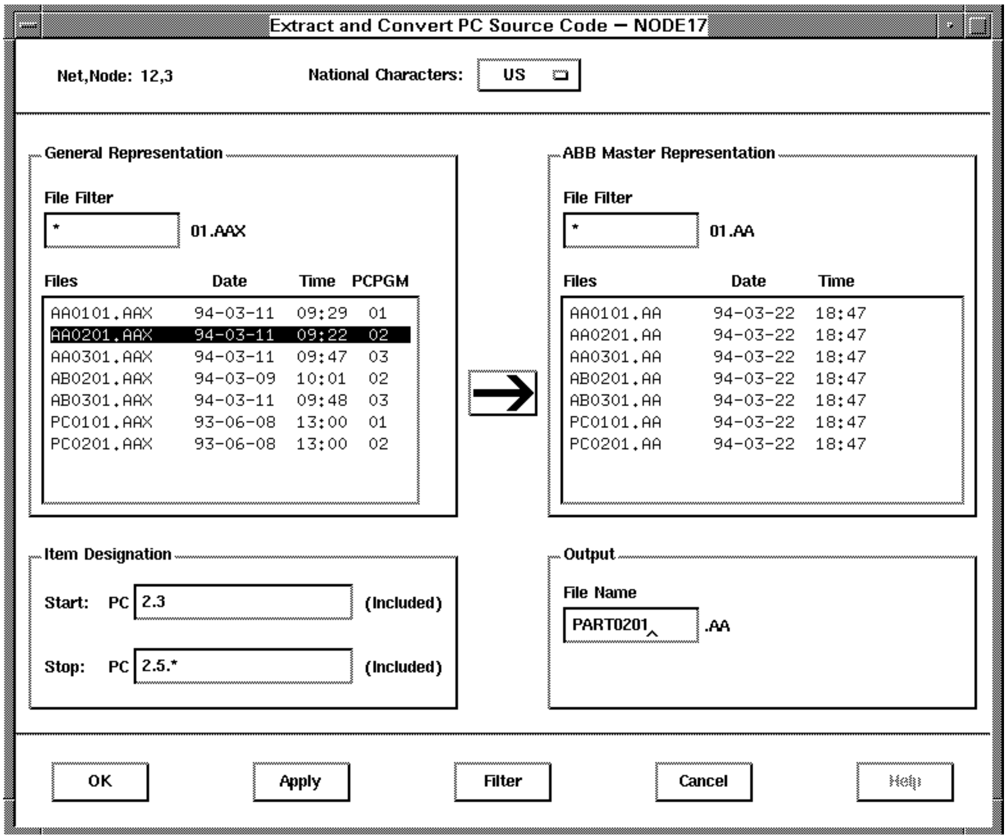
Keep in mind, that the directors of the old object are still selected in the On-line Builder. Up-loading or down-loading of source code files in this situation will result in wrong assignments. Therefore, use SLTARG for testing only, or even better, leave the On-line Builder, select the new object within the Structure Builder, then start the On-line Builder again.

Calling Commands while connected to node 0,0

If the network and node number shown on the title bar of the On-line Builder window is 0,0, then the On-line Builder is not connected to a real target. Most of the Commands will not work properly in this case. Connect to an existing target with the Command SLTARG or even better, leave the On-line Builder.

3.7.8 How to Extract PC Source Code

As a first step this function creates a smaller PC source code file out of a bigger one. That is to say, it extracts a part of a PC source code file and creates a new one. This new file will be prepared and transferred to the SRCE directory.



Extract and Convert PC Source Code – NODE17

Net,Node: 12,3

National Characters: US

General Representation

File Filter: * 01.AAX

Files	Date	Time	PCPGM
AA0101.AAX	94-03-11	09:29	01
AA0201.AAX	94-03-11	09:22	02
AA0301.AAX	94-03-11	09:47	03
AB0201.AAX	94-03-09	10:01	02
AB0301.AAX	94-03-11	09:48	03
PC0101.AAX	93-06-08	13:00	01
PC0201.AAX	93-06-08	13:00	02

ABB Master Representation

File Filter: * 01.AA

Files	Date	Time
AA0101.AA	94-03-22	18:47
AA0201.AA	94-03-22	18:47
AA0301.AA	94-03-22	18:47
AB0201.AA	94-03-22	18:47
AB0301.AA	94-03-22	18:47
PC0101.AA	94-03-22	18:47
PC0201.AA	94-03-22	18:47

Item Designation

Start: PC 2.3 (Included)

Stop: PC 2.5.* (Included)

Output

File Name: PART0201 .AA

Figure 3-286. Extract source code window

First, you have to select the file you want to prepare from a list of files. This file has to be marked. Additionally you have to specify an output file name which must not be identical with the input file name.

Afterwards you specify a start and stop PC item designation to define which part of the PC program you want to extract. Activate the function using <OK> or <Apply>.

The asterisk (*) is allowed as a wildcard to specify a stop item designation. It is allowed only once in the item designation and only if following the period character (.). The wildcard is useful if you do not know the exact number of PC elements under, for example, the PC element CONTRM.

Table 3-23. Examples of Extract and Prepare

Start	Stop	Part that is extracted
PC1	PC1.3	PC1 to PC1.3 including PC1.3
PC1.1	PC1.*	PC1.1 to enc of PC1
PC1.2	PC1.3.*	PC1.2 to the last element in PC1.3 (e.g. PC1.3.99)

Example:

Start item designation:PC1.2.1
Stop item designation:PC1.3.*

```
EXAM0101.AAX              PART0101.AA

BEGIN PC1BEGIN PC1
HEADERHEADER
                Extract and Prepare
                ──────────────────►

PC1  PC1.2.1
PC1.1.
PC1.1.1PC1.97
PC1.1.2.
.    PC1.3
PC1.2PC1.3.1
PC1.2.1PC1.3.1
.
PC1.97END PC1
.
PC1.3
PC1.3.1
PC1.4
.
END PC1
```

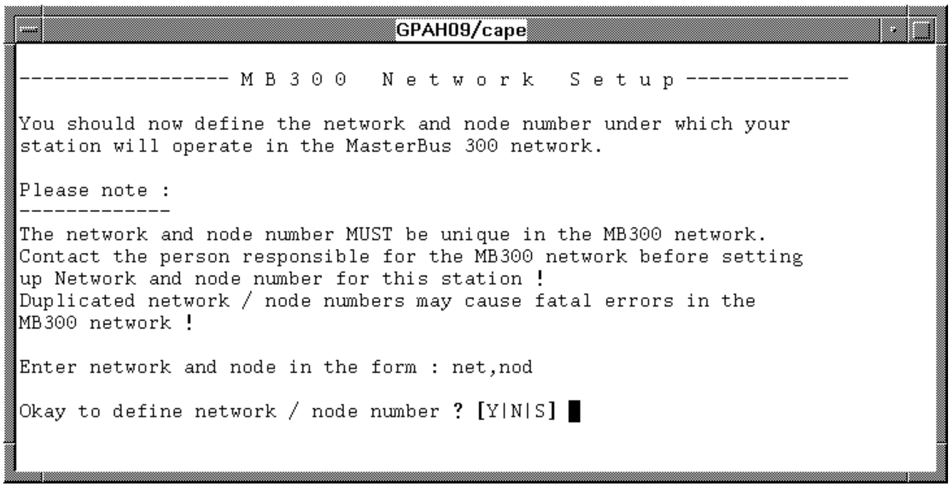
3.7.9 How to Change the Configuration of Net, Node

The Advant Station 500 Engineering Station may be connected to the MasterBus 300 via the RealTime Accelerator board. As a node of the MasterBus 300, the Engineering Station network and node numbers have to be defined. The installation of the On-line Builder and the first time definition of the network and node number is described in the Advant Station 500 Hardware documentation. This section provides a description of how to change network and node numbers, for example when you want to connect the Engineering Station to a different Master network.

The following procedure may be used to define both the network and node number and the printers. The printer definition part may be included or skipped, as required.

1. Physically disconnect the Engineering Station from the MasterBus 300.
2. Log in as user **root**.
3. Open a hp-terminal.
4. In the hp-terminal, type **/usr/local/bin/onf_config <Return>**.

5. Press the **Return** key. You will see a question as in Figure 3-287.

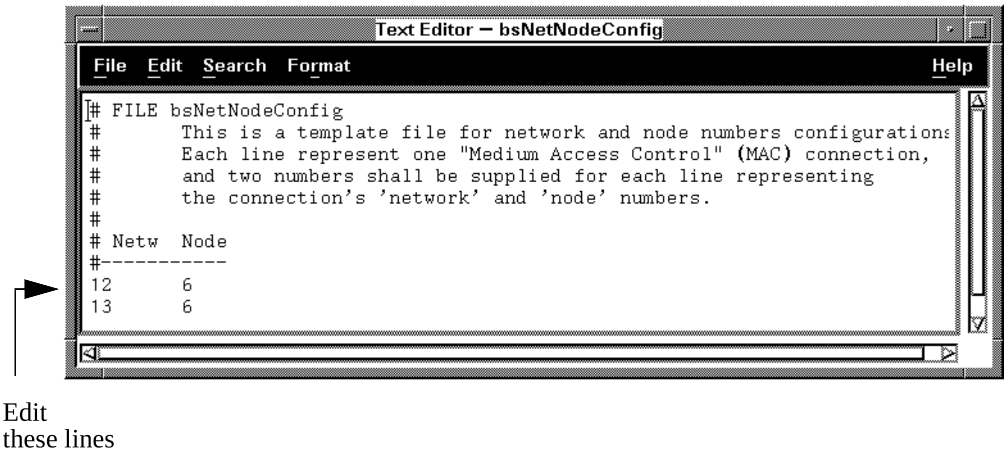


```
------------------ M B 3 0 0   N e t w o r k   S e t u p --------------

You should now define the network and node number under which your
station will operate in the MasterBus 300 network.

Please note :
-------------
The network and node number MUST be unique in the MB300 network.
Contact the person responsible for the MB300 network before setting
up Network and node number for this station !
Duplicated network / node numbers may cause fatal errors in the
MB300 network !

Enter network and node in the form : net,nod

Okay to define network / node number ? [Y|N|S] █
```

Figure 3-287. MB300 network setup

6. Type **Y <Return>** in order to define new network and node numbers. (Type **S <Return>** or **N <Return>** (or any other character) if you only want to define new printers, but don't want to change network and node numbers.)

If you answered **Y**, then you will now see the Text Editor window with the file **bsNetNodeConfig**, as in Figure 3-288.



```
# FILE bsNetNodeConfig
#       This is a template file for network and node numbers configurations
#       Each line represent one "Medium Access Control" (MAC) connection,
#       and two numbers shall be supplied for each line representing
#       the connection's 'network' and 'node' numbers.
#
# Netw  Node
#-----------
12      6
13      6
```

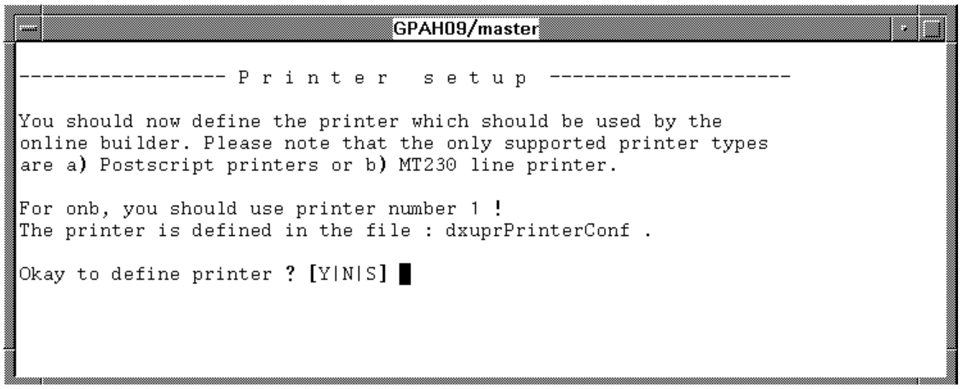
Figure 3-288. Edit network and node number

7. Change the network and node number of the Engineering Station by editing the first line, as indicated in Figure 3-288.

8. Change the network and node number of the Engineering Station on the (optional) redundant MasterBus 300 by editing the second line, as indicated in Figure 3-288.

9. Save the file and leave the editor.

In the hp-terminal, you will see a question as in Figure 3-289.



```
----------------- P r i n t e r   s e t u p ---------------------

You should now define the printer which should be used by the
online builder. Please note that the only supported printer types
are a) Postscript printers or b) MT230 line printer.

For onb, you should use printer number 1 !
The printer is defined in the file : dxuprPrinterConf .

Okay to define printer ? [Y|N|S] █
```

Figure 3-289. Printer setup

10. Type **Y <Return>** in order to define new printers (see section .... for details). Type **S <Return>** or **N <Return>** (or any other character) if you don't want to change the printer definitions.

If you answered **S**, then

11. Log out by clicking on the **Exit** symbol on the lower right of the screen.
12. Log in as user **master**.
13. Open a hp-terminal.
14. In the hp-terminal, type **$C_BIN_ONF_ETC/rta_boot <Return>**.
15. Answer the question by typing **Y** (without <Return>). After some seconds, you will get a message reading 'BOOT OK'.
16. Continue by pressing the **Return** key.
17. Physically connect the Engineering Station to the new MasterBus 300.

3.7.10 How to Change the Configuration of Printers

In order to have printouts from the On-line Builder, printers have to be defined for the On-line Builder. The installation of the On-line Builder and the first time definition of printers is described in the Advant Station 500 Hardware documentation. In this section, it is assumed that at least one printer has already been defined correctly.

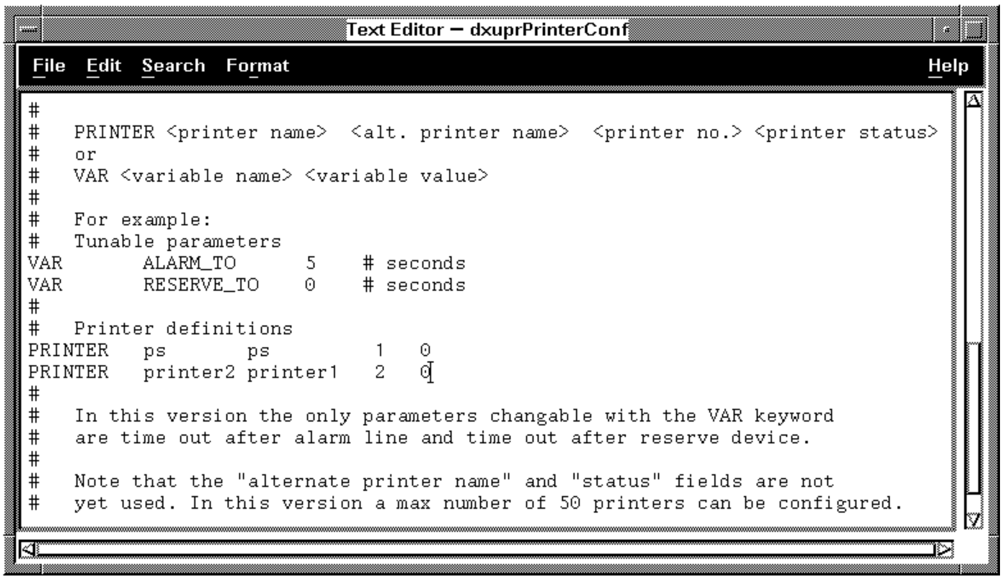
The following procedure may be used to define both the network and node number and the printers. See section 5.2 if you want to change the network and node number, too.

1. Log in as user **root**.
2. Open a hp-terminal.
3. In the hp-terminal, type **/usr/local/bin/onf_config <Return>**.
4. Press the **Return** key. You will see a question as in Figure 3-287.
5. Type **S <Return>** or **N <Return>** if you want to define new printers, but don't want to change network and node numbers.

If you answered **S**, then in the hp-terminal you will see a question as in Figure 3-289.

6. Type **Y <Return>** in order to define new printers.
(Type **N <Return>** (or any other character) if you want to quit the whole procedure.)

If you answered **Y**, then you will now see the Text Editor window with the file **dxuprPrinterConf**, as in Figure 3-290.



```
#
#   PRINTER <printer name>  <alt. printer name>  <printer no.> <printer status>
#   or
#   VAR <variable name> <variable value>
#
#   For example:
#   Tunable parameters
VAR       ALARM_TO      5    # seconds
VAR       RESERVE_TO    0    # seconds
#
#   Printer definitions
PRINTER   ps       ps         1   0
PRINTER   printer2 printer1   2   0
#
#   In this version the only parameters changable with the VAR keyword
#   are time out after alarm line and time out after reserve device.
#
#   Note that the "alternate printer name" and "status" fields are not
#   yet used. In this version a max number of 50 printers can be configured.
```

Edit
these lines

Figure 3-290. Edit printer definitions

7. Edit the lines beginning with the keyword **PRINTER** as explained in the file.
8. Save the file and leave the editor.
9. Log out by clicking on the **Exit** symbol on the lower right corner of the screen.
10. Log in as user **master**.
11. Open a hp-terminal.
12. In the hp-terminal, type **$C_BIN_ONF_ETC/rta_boot <Return>**.
13. Answer the question by typing **Y** (without <Return>). In a few seconds, you will see a message reading 'BOOT OK'.
14. Continue by pressing the **Return** key.

3.7.11 How to Use the On-line Builder with a MP51

To work with a MasterPiece 51 the key definition of the keyboard must be changed. The user must change it by himself. The current key table must be saved, the MP51 key table loaded and after the MP51 working session the saved key table must be loaded again.

There is a file /products/components/onf/bin/mp51_keys.sh which saves and loads the current key table:

```
mp51_keys.sh[ -h(show help informations) ]
            [ -r     (read key file) ]
            [ -f  file(file with key table for xmodmap) ]
            [ -u     (read file vt100keypad with MP51 key table) ]
```

Use a script as shown in the following steps:

1. Execute
   ```
   mp51_keys.sh
   ```
   without parameters. It save the current key table to $HOME/tmp/mp51_keys_old.lis . If you want to give an own filename, use 'mp51_keys.sh -f filename'.

2. Execute
   ```
   mp51_keys.sh -u
   ```
   to read file vt100keypad with MP51 key table.

3. After MP51 working execute
   ```
   mp51_keys.sh -r
   ```
   to read saved key table from $HOME/tmp/mp51_keys_old.lis. Have you saved the current key table in an own file, use 'mp51_keys.sh -r -f filename' to load your own file.

CAUTION

It is important to load the original key table after MP51 working. Otherwise the keyboard works wrong with other applications.

3.7.12 Central Backup and Restore

3.7.12.1 How to Backup a Complete Plant

To up-load memory dumps of all selected target nodes execute the program:

dobackup *project net,node* **{** *net,node* **}**

Examples:
backup of 1 node: "dobackup mas_exp 12,2"
backup of 4 nodes: "dobackup mas_exp 12,2 12,3 12,4 12,5"

The dobackup will create a new directory below the directory of

/proj/*projectname***/nodes/backup.**

The name of the directory will be a combination of the date and the time when it is created (e.g. 940411-17:00).

The file names within the directory will be a combination of network and node number, a counter and the string '.AD'.
For example: 01200201.AD is an application dump of node 2 on network 12.

3.7.12.2 How to Restore a Complete Plant

To down-load memory dumps of all selected target nodes which had been saved on the hard disk before execute the program:

```
dorestore[-p | -d project | -n  project date_directory ]  \
            [ project date_directory net,node { net,node } ]

            -p                ... show projects
            -d project        ... show date_directories
            -n project date_directory... show net, node numbers
```

Examples:
restore of 1 node:“dorestore mas_exp 940411-17:00 12,2”
restore of 4 nodes:“dorestore mas_exp 940411-17:00 12,2 12,3 12,4 12,5”

3.8 How to Edit PC and DB Source Code

Source code files may be edited with a text editor.

In the Structure Builder tree presentation select the target object you want to edit source code. Choose **Object ‘Special Commands’**. The Special Commands dialog box will pop up. Click on the **FILE** entry in the list. Then click on the **OK** button.

You will see a window, which displays the directory of the selected object (as in Figure 3-291). Source code files will be found in the directors of DBDATA, PCDATA, TCDATA and SRCE or USER.

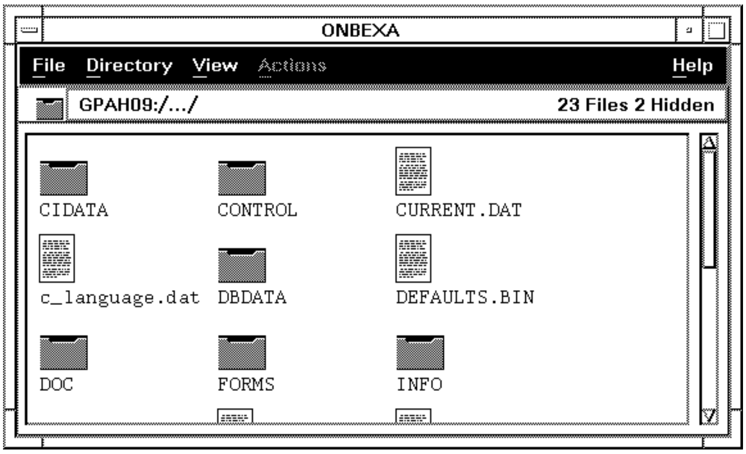
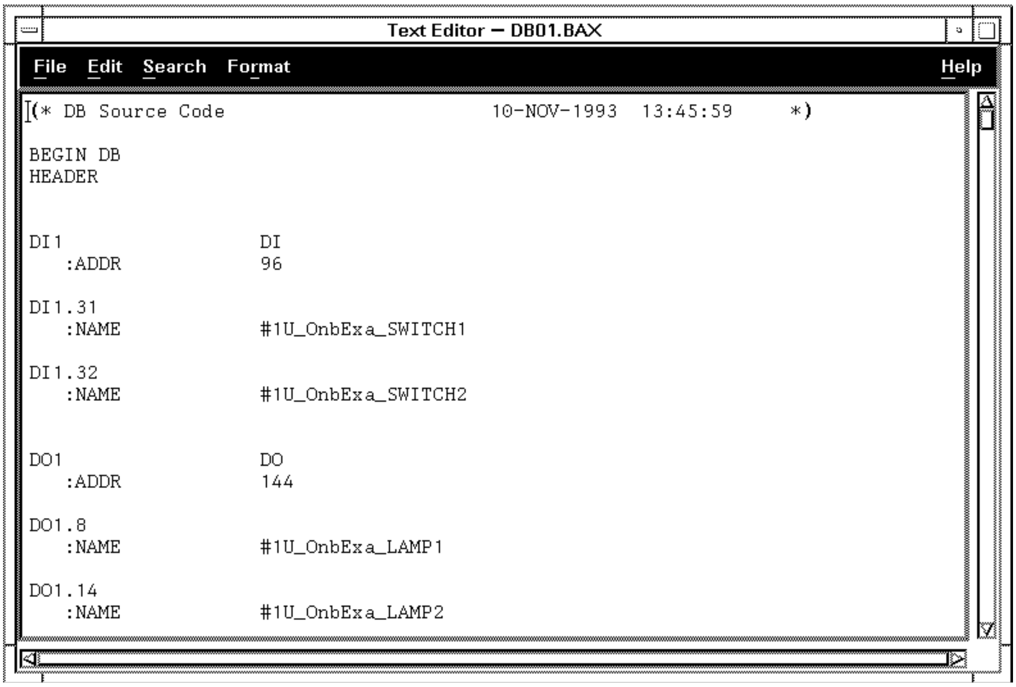


Figure 3-291. VUE file manager window

1. Double click on one of those symbols in order to open that subdirectory. You will see a list of source code files. (Double click on the symbol **..(go up)** if there is no suitable file to edit.)
2. Double click on the file you want to edit. In the Text Editor Window (as in Figure 3-292), you will see the selected file.



```
(* DB Source Code                         10-NOV-1993  13:45:59     *)

BEGIN DB
HEADER


DI1                 DI
   :ADDR            96

DI1.31
   :NAME            #1U_OnbExa_SWITCH1

DI1.32
   :NAME            #1U_OnbExa_SWITCH2


DO1                 DO
   :ADDR            144

DO1.8
   :NAME            #1U_OnbExa_LAMP1

DO1.14
   :NAME            #1U_OnbExa_LAMP2
```

Figure 3-292. VUE Pad Editor

3. Edit the file. Consult the HP Visual User Environment User's Guide about how to use the text editor.
4. Leave the text editor. Click on **File** in the menu bar of the Text Editor window, then click on **Exit** in the File menu.
5. In the dialog box, click on the **Yes** button in order to save the modifications.
   (Click on the **No** button if you want to leave the editor without saving.)
6. Click on **File** in the menu bar of the window showing the directory, then click on **Close** in the File menu.

Chapter 4 Runtime Operation

4.1 Product Operation

This section is not applicable.

4.2 Operating Overview

This section is not applicable.

4.3 Runtime Tutorial

This section is not applicable.

4.4 Operating Instructions

This section is not applicable.

4.5 Runtime Operation Menus

This section is not applicable.

Chapter 5 Maintenance

5.1 Error Messages

5.1.1 System Administration Error Messages

5.1.1.1 Location of Error Messages

Error messages of the Administration component are not displayed via dialog boxes as with other tools but are written to the logfiles produced by the respective Administration programs. When you execute an administration function, if an error occurs, you are notified that an error occurred by an information box which recommends that you look at the logfile for further information).

5.1.1.2 General Error Messages

Error message: %ERROR : not user <user_name>

Cause: This error occurs when you log into STB as admin user to start Administration with an operating system user account other than *cape*.

Action: Make sure you are logged in as the *cape* operating system user before you start the STB and log in as the *admin* user.

Error message: %ERR : shellscript / sqlscript <file_name> missing

Cause: The shellscript / sqlscript <file_name> needed by the program is missing or cannot be read or executed.

Action: This problem should not occur since all required files should be on the installation tape from which your production environment was created. Check if the file is present and accessible for the system user *cape*. If the file is missing, contact your system administrator to recreate it from tape.

Error message: %ERROR : environment variable <env_var> not set

Cause: All environment variables should be set for the user automatically. If this error occurs, something is wrong with your environment.

Action: Log out and in again, then try again.

5.1.1.3 Error Messages Caused by Concurrent Execution of Administrative Tasks

The following error messages may occur during execution of any Administration function such as when you insert a project or user, save or restore a project and so on. To find out which state a project is in, press **<Execute Query>** in the Projects window and check the status column.

Error message: %ERROR : project is being generated, aborting!

Cause: The project you want to manipulate via an Administration menu item is just being generated by the insert project function and not yet complete.

Action: Wait until the insert project function is finished and try again.

Error message: %ERROR : project is being copied from, aborting!

Cause: The project you want to manipulate via an Administration menu item is just being copied by the insert project function and should not be modified.

Action: Wait until insert project function is finished and try again.

Error message: %ERROR : project is being deleted, aborting!

Cause: The project you want to manipulate via an Administration menu item is just being deleted so it does not make sense to invoke any administration function.

Action:

Error message: %ERROR : project is being saved, aborting!

Cause: The project you want to manipulate via an Administration menu item is just being saved and should not be modified during this process.

Action: Wait until project save is finished and try again.

Error message: %ERROR : project is being restored, aborting!

Cause: The project you want to manipulate via an Administration menu item is just being restored and should not be accessed during this process.

Action: Wait until the restore project function is finished and try again.

Error message: %ERROR : project is being enlarged, aborting!

Cause: Occurs only when trying to enlarge or delete a project. The project you want to manipulate is either already being enlarged and no further enlargement is necessary or it is assumed that a project which is being enlarged is still needed and therefore should not be deleted.

Action: Wait until the enlarge project function has finished. Make sure you really want to enlarge this project further (refer to size fields in Projects window).

Error message: %ERROR : project is being modified, aborting!

Cause: The project you want to manipulate via an Administration menu item is just being modified and should not be accessed during this process.

Action: Wait until the modify project function is finished and try again.

Error message: %ERROR : undefined status, aborting!

Cause: Invalid project status, this error should not occur.

Action: Use script set_project_state.sh to set project status to 'A' (see Section 3.4.1.10, How to Handle a Corrupted Project).

Error message: %ERROR : project is being generated / deleted / restored / saved - no user addition/deletion/modification allowed, aborting!

Cause: Project is being modified by an administrative task. During this process no user should be inserted, deleted or modified.

Action: Wait until the project is accessible again (as indicated by A in Status column) and try again.

5.1.1.4 Error Messages Caused by 'Insert / Delete Project'

Error message: %ERROR : source project probably being modified, aborting!

Cause: Some users are working on the project which is to be used as source for a new project. It is assumed that the source project is modified and it does not make sense to copy it.

Action: Make sure nobody is working on the source project and try again.

Error message: %ERROR : tablespace ES_<project_name> already exists, exiting!

Cause: This error should never occur. It means that a project of the chosen name already exists. This should have already been checked when you entered the project name in the Insert Project dialog box.

Action: Choose another name for your project.

Error message: %ERROR : project probably being modified, aborting!

Cause: This error message occurs when trying to delete a project on which users are still working. It is assumed that a project somebody is working on is still required.

Action: Make sure nobody is working on the project in question and try again.

Error message: %ERROR : probably not possible to drop all project tables, aborting!

Cause: This error message occurs when trying to delete a project that users are still working on, thereby causing database locks on tables which, during the delete project function, are supposed to be dropped.

Action: Make sure nobody is working on the project and try again.

Error message: %ERROR : occurred in: checking available disc space, see logfile credir_*.log!

Cause: Before creating the directories for a new project, the available disc space is checked by a special shellscript which writes it's messages to file credir_<project_name>.log. The error message is probably caused by not having enough free disc space available.

Action: Check the logfile credir_<project_name>.log for lines containing further error messages, see Section 5.1.1.8, Error Messages Caused by Disc Space Check during Insert/Save/Enlarge Project.

Error message: %ERROR : ORA-1547: failed to allocate extent of size <size> in tablespace <tablespace_name>

Cause: Might occur when you insert a new project where <tablespace_name> is SYSTEM. It is also possible that this message is truncated so that <tablespace_name> is not shown. This message is caused by the ORACLE data dictionary space being too small to accommodate the storage of the names of the tables and indexes created for the new project.

Action: Check the available space of the ORACLE data dictionary by entering sqlplus interactively as user *admin* and typing:

```
SELECT SUM(BYTES) FROM DBA_FREE_SPACE
WHERE TABLESPACE_NAME = ‘SYSTEM’;
```

If result < 300 000 (= 3 Megabyte), then find out first where the data file(s) for the ORACLE data dictionary reside(s):

```
SELECT FILE_NAME FROM DBA_DATA_FILES
WHERE TABLESPACE_NAME = ’SYSTEM’;
```

The result will look like
<pathname>/<file_name>_**<number>**.dbf, e.g.
*/proj/proj_adm/adm/dat/ES_DB_system_**1**.dbf.*

Add 10 Megabytes to the ORACLE data dictionary by typing:

```
ALTER TABLESPACE SYSTEM
ADD DATAFILE ‘<additional_data_file_name>’ SIZE 10M;
```

where <additional_data_file_name> is

<pathname>/<file_name>_**<highest_number_plus_one>**.dbf, e.g.
*/proj/proj_adm/adm/dat/ES_DB_system_**2**.dbf.*

5.1.1.5 Error Messages Caused by ‘Enlarge Project’

Error message: %ERR : not enough free space on disc, aborting!

Cause: Before enlarging a project, the available disc space is checked by a special shellscript which writes it’s messages to file check_dir_<project_name>.log. The error message is probably caused by not having enough free disc space available to add another datafile to the project.

Action: Check the logfile check_dir_<project_name>.log for lines containing further error messages, see Section 5.1.1.8, Error Messages Caused by Disc Space Check during Insert/Save/Enlarge Project.

5.1.1.6 Error Messages Caused by ‘Save Project’

Error message: %ERR : not enough free space to save directories

Cause: Before enlarging a project, the available disc space is checked by a special shellscript which writes it’s messages to file check_dir_<project_name>.log. The error message is probably caused by not having enough free disc space available to add another datafile to the project.

Action: Check the logfile check_dir_<project_name>.log for lines containing further error messages, see Section 5.1.1.8, Error Messages Caused by Disc Space Check during Insert/Save/Enlarge Project.

Error message: %ERROR : temp file <file_name> not readable, aborting

Cause: When checking the available disc space, a temporary file is created in the directory the environment variable $C_BIN_CAPE_TMP points to and later removed again. If it was not possible to create this file, the above error message results. This should not happen unless the protection of $C_BIN_CAPE_TMP has been changed manually. Another possible reason could be too many processes for user *cape* are running.

Action: Check if directory $C_BIN_CAPE_TMP is writable for user *cape* and is empty, then try again. Make sure you don’t have too may processes running to possibly exceed your quota.

Error message: %ERR : error occurred during file copy

Cause: When copying the project directory subtree, this error might occur, for example, when cape user has excised his process quota.

Action: Check logfile cop_subtree_<project_name>_<release_name>.log in directory $C_CAPE_ALL_ADM_LOG and try again.

Error message: %ERR: error during removal of original export file

Cause: After you save the complete project directory subtree, the original database export file in directory $C_CAPE/<project_name>/sav is removed by an operating system command. When the process quota for the cape user is exhausted, this removal might fail with the above message.

Action: If this is the only error that occurred during the project save, it can be ignored with the result that the original export file in $C_CAPE/<project_name>/sav still exist. The other possibility is to remove the file named <project_name>_<release_name>.exp manually.

5.1.1.7 Error Messages Caused by ‘Restore Project’

Error message: %ERR : allocation of memory for username failed, exiting!

Cause: Internal program error.

Action: Try again. In case of repeated failure contact your support.

Error message:	**%ERR : export-file <file_name> missing!**
Cause:	The database export file which during the save process was written to directory *sav* underneath the user chosen save directory, does not exist anymore.
Action:	Check if the file exists and is readable for *cape*. If not, ask your system administrator if a backup of this file exists, copy it to the *sav* directory shown in the error message and start the program again.

5.1.1.8 Error Messages Caused by Disc Space Check during Insert/Save/Enlarge Project

The disc space available is checked when:

- inserting a new project (logfile name credir_<project_name>.log)
- saving a project (logfile name check_dir_<project_name>.log) or
- enlarging a project (logfile name check_dir_<project_name>.log).

When Insert Project is chosen, the space on the disc on which directory $C_CAPE is located, is checked. If there is not enough space available, all other available discs are checked.

When Save Project is chosen, the space on the disc on which the user chosen save directory is located, is checked. If there is not enough space is available on this disc, try a directory on another disc with enough space.

When Enlarge Project is chosen, the space on the disc on which directory $C_CAPE/<project_name> is located, is checked.

Error message:	**%ERROR : <project_name> project is not available. Exiting ...**
Cause:	The existence of directory $C_CAPE/<project_name> is checked for which the physical path of $C_CAPE is read from a system file. If the entry for $C_CAPE in this system file does not conform to the actual physical path of $C_CAPE, this error message occurs.
Action:	Contact your support, include a copy of file */abbinstall/setup.dat.*

Error message:	**%ERROR : disc named <disc_name> is unknown. Exiting ...**
Cause:	Internal error.
Action:	Contact your support, include a copy of file */abbinstall/setup.dat.*

Error message:	**%ERROR : <system_file_name> not readable or missing. Exiting ...**
Cause:	The system file <system_file_name> has a wrong protection or is missing.
Action:	Check if cape user can read this file. If it is missing, contact your support.

Error message:	**%ERROR : disc <disc_name> does not have enough space available...**
Cause:	No space left on disc.
Action:	Ask system administrator to clean up the disc to gain some more space or choose a smaller size for a new project. The additional datafile of a project. For the project save, you may also choose a save directory on another disc.

Error message: **%ERROR : no space on <disc_name> to enlarge / save project.**

Cause: No space left on disc.

Action: Ask system administrator to clean up the disc to gain some more space or choose a smaller size for the additional datafile if the project is to be enlarged. In case of project save you may also choose a save directory on another disc.

Error message: **%ERROR : no disc has enough space available. Exiting ...**

Cause: No space left on any disc.

Action: Ask your system administrator to clean up discs if possible.

Error message: **%ERROR : savedir path named <path_name> not possible, no disc....**

Cause: The savedir pathname given in the dialog box for Save Project does not pertain to an existing disc.

Action: Make sure the pathname starts at least with an existing disc.

5.1.1.9 Error Messages Caused by 'Insert / Delete User'

Error message: **%ERROR : username <user_name> should start with letter and continue with letters, digits or '_', aborting!**

Cause: You entered an invalid sequence of alphanumeric tokens for the username.

Action: A username must always start with a letter and can contain only letters, digits or '_' (underscore).DO NOT use any special characters like '@' or '$'.

Error message: **%ERROR : password <password> should start with letter and continue with letters, digits or '_', aborting!**

Action: same as above.

Error message: **%ERROR : deletion of <user_name> not allowed, aborting!**

Cause: You tried to delete the user <project_name>_CFG.

Action: This user cannot be deleted.

5.1.2 Structure Builder Error Messages

Error message: TSH: 100: The specified object type '<OBJ_TYPE>' cannot be used as parent object type. Use another one.

Cause: You are not allowed to add a child object to the selected object type.

Action: Use another object type for the parent.

Error message: TSH: 102: Cannot perform this action because the parent object id is empty. Enter a valid parent object id and try again.

Cause: You are not allowed to add a child object to the selected object type.

Action: Use another object type for the parent.

Error message: TSH: 104: Cannot perform this action because the object id is empty. Enter a valid object id and try again.

Cause: The object id is not set.

Action: Select or enter a valid object id.

Error message: TSH: 106: Cannot perform this action because the object type is empty. Enter a valid object type and try again.

Cause: The object type is not set.

Action: Select or enter a valid object type.

Error message: TSH: 108: Internal error. Cannot add new aspect window of ‘<OBJ_ID>’ in aspect window array. Please contact your system administrator.

Cause: An internal software error occurred (for example, memory allocation).

Action: Contact your system administator.

Error message: TSH: 110: Internal error. Cannot find the aspect window of display object ‘<OBJ_ID>’. Please contact your system administrator.

Cause: An internal software error occurred.

Action: Contact your system administator.

Error message: TSH: 112: Cannot perform this action because the typical solution id is empty. Select one and try again.

Cause: The solution object id is not set.

Action: Select one and try again.

Error message: HLC: 100: Cannot find the top project with parent tree id of ‘0’. Check the project data base installation.

Cause: The project object was not found.

Action: Check the project data base installation or contact your system administator.

Error message: HLC: 102: Cannot find the object ‘<OBJ_ID>’ in the project data base. Check the object id.

Cause: The specified object id was not found in the project.

Action: Check the spelling of id and try again.

Error message: HLC: 104: Internal error. Cannot find the parent of object with the tree id ‘<TREE_ID>’. Check the tree id and call your system administrator.

Cause: The specified object id was not found in the project.

Action: Check the spelling of id and try again.

Error message: HLC: 106: Cannot perform this action because the child sequence number of object '<OBJ_ID>' was not found in the project data base. Check the object id.

Cause: The specified object id was not found in the project.

Action: Check the spelling of id and try again.

Error message: HLC: 108: Cannot perform this action because the parent of object '<OBJ_ID>' was not found in the project data base. Check the object id.

Cause: The specified object id was not found in the project.

Action: Check the spelling of id and try again.

Error message: HLC: 110: Cannot perform this action because the tree id of object '<OBJ_ID>' was not found in the project data base. Check the object id.

Cause: The specified object id was not found in the project.

Action: Check the spelling of id and try again.

Error message: HLC: 112: Cannot perform this action because the object '<OBJ_ID>' was not found in the project data base. Check the object id.

Cause: The specified object id was not found in the project.

Action: Check the spelling of id and try again.

Error message: HLC: 114: Cannot perform this action because the object '<OBJ_ID>' was not found in the project '<TAB_OWNER>' data base. Check the object id and project name.

Cause: The specified object id was not found in the specified project.

Action: Check the spelling of id and project name and try again.

Error message: HLC: 116: Cannot perform this action because comment and parent tree id of the object '<OBJ_ID>' was not found in the project data base. Check the object id and try again.

Cause: The specified object id was not found in the project.

Action: Check the spelling of id and try again.

Error message: HLC: 118: Cannot perform this action because the '<LINK_TYPE>' link of object '<OBJ_ID>' is not correct. Check the object id and the correct link specification.

Cause: An invalid link type was used.

Action: Check the object id and link type and try again.

Error message: HLC: 120: Cannot perform this action because the description of solution '<OBJ_ID>' was not found in the project data base. Check the object id.

Cause: The description of an object was not found in the project.

Action: Check the spelling of id and try again.

Error message: HLC: 150: Cannot move object '<OBJ_ID>' to the parent '<PAR_OBJ_ID>' because it is the own sub tree. Use another parent object and try again.

Cause: An object was moved to its own subtree. This is not allowed.

Action: Take another parent object outside the selected subtree and try again.

Error message: HLC: 152: Internal error. Cannot '<COMMAND>' object '<OBJ_ID>' because the category is not 'objects'.

Cause: The category of an object was mismatched.

Action: Contact your system administrator.

Error message: HLC: 154: Cannot '<COMMAND>' object '<OBJ_ID>' because the type '<OBJ_TYPE>' does not match to the parent type '<PAR_OBJ_TYPE>'. Use another destination object type.

Cause: The specified object type cannot be put under the parent type.

Action: Use another destination object type and try again.

Error message: HLC: 156: Cannot '<COMMAND>' object '<OBJ_ID>' because the new parent object '<NEW_OBJ_ID>' was not found. Check parent id.

Cause: The parent object id was not found in the project.

Action: Check the spelling of the id and try again.

Error message: HLC: 158: Cannot '<COMMAND>' object '<OBJ_ID>' because the new object id '<NEW_OBJ_ID>' is too long. Check object id.

Cause: The specified object id was not found in the project.

Action: Check the spelling of the id and try again.

Error message: HLC: 160: Cannot '<COMMAND>' the comment '<COMMENT>' because the new comment '<NEW_COMMENT>' is too long. Check object comment.

Cause: The specified comment was too long.

Action: Check length of comment and try again.

Error message: HLC: 162: Cannot perform this action because the link type '<LINK_TYPE>' is not valid. Check the link type.

Cause: The specified link type was not valid.

Action: Check the spelling and try again.

Error message: HLC: 190: Missing selected object in data base.

Cause: The specified object id was not found in the project.

Action: Check the spelling of id and try again.

Error message: HLC: 191: The object is locked by another user.

Cause: The object is locked by another user.

Action: Try again or call your system administration.

Error message: HLC: 192: Links to the selected element are locked by another user.

Cause: Links of an object cannot be changed because they are locked by another user.

Action: Try again or contact your system administrator.

Error message: HLC: 193: Subordinated elements are locked by another user.

Cause: The object and its subtree are locked by another user and cannot be used now.

Action: Try again or contact your system administrator.

Error message: HLC: 194: Links to subordinated elements are locked by another user.

Cause: Links of an object cannot be changed because they are locked by another user.

Action: Try again or contact your system administrator.

Error message: HLC: 195: Cannot access to all subordinated elements.

Cause: Some objects in the subtree are not accessible.

Action: Try again or contact your system administrator.

Error message: HLC: 196: Subordinated elements are locked by another user.

Cause: The object and its subtree are locked by another user and cannot be used now.

Action: Try again or contact your system administrator.

Error message: STB: 100: Internal error. Cannot find parent object of '<OBJ_ID>' with tree id '<TREE_ID>'. Check spelling of object id and tree id. Please contact your system administrator.

Cause: An internal software error occurred (inconsistent data).

Action: Contact your system administrator.

Error message: STB: 102: Internal error. Cannot find the project object because it is NULL. Please contact your system administrator.

Cause: An internal software error occurred (inconsistent data).

Action: Contact your system administrator.

Error message: STB: 104: Cannot move up to the project object because it is already set.

Cause: The specified object is already on the top.

Action: No action is necessary.

Error message: STB: 106: There is no object selected. Select one object to perform this action.

Cause: To perform this action one object has to be selected.

Action: Select one object and try again.

Error message: STB: 108: There are several objects selected. Select only one object to perform this action.

Cause: To perform this action one object has to be selected.

Action: Select one object and try again.

Error message: STB: 110: Cannot start the following tool with command '<SYS_COMMAND>'. Look also at the current terminal window output. Please contact your system administrator.

Cause: The system cannot perform this action.

Action: Contact your system administrator.

Error message: STB: 111: Internal error. Cannot start menu action. There is an unrecognized switch variable. Please contact your system administrator.

Cause: An internal software error occurred (inconsistent data).

Action: Contact your system administrator.

Error message: STB: 120: Internal error '<RTC_SUBJECT>':'<RTC_CODE>'. Cannot read objects from data base. Please contact your system administrator.

Cause: An internal software error occurred.

Action: Contact your system administrator.

Error message: STB: 122: Internal error. Cannot execute a query in the list window of '<OBJ_ID>' because the last query conditions were not found. Please contact your system administrator.

Cause: An internal software error occurred (inconsistent data).

Action: Contact your system administrator.

Error message: STB: 124: Internal error. Cannot use display object of list window because it is NULL. Please contact your system administrator.

Cause: An internal software error occurred (inconsistent data).

Action: Contact your system administrator.

Error message: STB: 126: Internal error. Cannot use list window because window id is NULL. Please contact your system administrator.

Cause: An internal software error occurred (inconsistent data).

Action: Contact your system administrator.

Error message: STB: 128: Internal error. Cannot delete object because id or type is NULL. Please contact your system administrator.

Cause: An internal software error occurred (inconsistent data).

Action: Contact your system administrator.

Error message: STB: 130: The parent object of '<OBJ_ID>' was not found. Check id of object and parent.

Cause: The specified object id was not found in the specified project.

Action: Check the spelling of the id and try again.

Error message: STB: 132: Cannot find objects because id or type filter '<ID_FILTER>', '<TYPE_FILTER>' is empty. Use wildcard % for searching all objects.

Cause: The specified filters are not set.

Action: Use wildcard % to match all objects.

Error message: STB: 134: The Objects with the current id and type filter '<ID_FILTER>' - '<TYPE_FILTER>' were not found. Check the filter entries and try again.

Cause: The specified filters for object id and type are doesn't match any objects.

Action: Check the filter search pattern and try again.

Error message: STB: 136: Cannot set temporary root because the new temporary root object id is empty. Enter an existing object id and try again.

Cause: The object id was not set.

Action: Specify an object id and try again.

Error message: STB: 138: Cannot insert an object to the parent object type '<OBJ_TYPE>'. Change insert mode (on same level) or select a new parent and try again.

Cause: The parent object type cannot used for inserting.

Action: Use another one.

Error message: STB: 140: Cannot insert an object '<OBJ_TYPE>' on same level while its parent is a temporary root or is the top project. If it is the temporary root then move up one level and try again.

Cause: This action cannot be done on a temporary root or the top project.

Action: While the selected object is a temporary root navigate up and try again.

Error message: STB: 142: The expected dialog box is already displayed. Use the existing one or close and open it again.

Cause: The dialog box is already displayed.

Action: Use the existing one.

Error message: STB: 144: There is no object selected. Select one or more objects to preform this action.

Cause: To perform this action an object has to be selected.

Action: Select an object and try again.

Error message: STB: 146: The expected list window of the object '<OBJ_ID>' is already displayed. Use the existing one.

Cause: The list window is already displayed.

Action: Use the current one.

Error message: STB: 148: Internal error. Cannot insert new list window of '<OBJ_ID>' in list window array. Please contact your system administrator.

Cause: There is an internal software error (inconsistent data)

Action: Contact your system administrator.

Error message: STB: 150: Cannot use the substitute mode because the substitute field is empty. Enter a search pattern and try again.

Cause: The input field was empty.

Action: Enter a search pattern and try again.

Error message: STB: 152: Internal error. Cannot use the selected display object because it is NULL. Please call your system administrator.

Cause: There is an internal software error (inconsistent data)

Action: Contact your system administrator.

Error message: STB: 154: Cannot use the object '<OBJ_ID>' to perform this action because it is a temporary root or the top project. If it is a temporary root object then move up one level and try again.

Cause: This action cannot be done on a temporary root or the top project.

Action: While the selected object is a temporary root navigate up and try again.

Error message: STB: 156: Cannot move object because new parent '<PAR_OBJ_ID>' and old parent '<OBJ_ID>' are the same. Check parent objects and try again.

Cause: An attempt was made to move an object.

Action: Check the spelling of the id and project name and try again.

Error message: STB: 158: Internal error. Cannot use the parent of the selected display object '<OBJ_ID>' because it is NULL. Please call your system administrator.

Cause: There is an internal software error (inconsistent data)

Action: Contact your system administrator.

Error message: STB: 160: Cannot execute a special command because there are no special commands defined for the selected object type '<OBJ_TYPE>'.

Cause: There are no special commands defined for the specified object type.

Action: Use another object type and try again.

Error message: STB: 162: Cannot perform a '<COMMAND>' command because there are no view commands defined for the selected object type '<OBJ_TYPE>'.

Cause: There are no views defined for the specified object type.

Action: Use another object type and try again.

Error message: STB: 164: Cannot perform a '<COMMAND>' command because the object type is not set.

Cause: The object type was not set to perform this action.

Action: Check the object type and try again.

Error message: STB: 166: Cannot perform a '<COMMAND>' command because there are no object types defined for this command.

Cause: There are no commands defined for the specified object type.

Action: Use another object type and try again.

Error message: STB: 168: Cannot find other projects from where you can import objects. Check if there are other projects on the system available and accessible. Please call your system administrator.

Cause: No other projects are accessible from the data base.

Action: Check project data base or contact your system administrator.

Error message: STB: 172: Cannot find link types for the selected object type '<OBJ_TYPE>' because there are none defined. Use another object.

Cause: The specified object has no links.

Action: Use another one and try again.

Error message: STB: 174: In this mode no link types will be displayed. The current mode is reseted.

Cause: A mode was used where no link types are displayed.

Action: No action is necessary.

Error message: STB: 176: Cannot perform this action because the object type is not set. Check object type and try again.

Cause: The specified object type was not set.

Action: Check object type and try again.

Error message: STB: 180: Internal error. Cannot insert display object '<OBJ_ID>' because object already exists and has the parent tree id '<TREE_ID>'. Please call your system administrator.

Cause: An internal software error was occurred (inconsistent data)

Action: Contact your system administrator.

Error message: STB: 182: Internal error. Cannot insert display object '<OBJ_ID>' because object already exists. Please call your system administrator.

Cause: An internal software error occurred (inconsistent data)

Action: Contact your system administrator.

Error message: STB: 184: Cannot display more than '<TREE_COUNT>' tree objects. Reading is aborted. Use the temporary root function and try again.

Cause: You tried to display more than 500 objects in the tree window.

Action: Use temporary root function to reduce the number of displayed objects.

Error message: STB: 186: Cannot write to temporary file. Look at messages of the terminal window. Please call your system administrator.

Cause: The system cannot perform this action.

Action: Contact your system administrator or try again.

Error message: STB: 188: Cannot flush file '<FILE_NAME>'. Look at messages of the terminal window. Please call your system administrator.

Cause: The system cannot perform this action.

Action: Contact your system administrator or try again.

Error message: STB: 190: Cannot close file '<FILE_NAME>'. Look at messages of the terminal window. Please call your system administrator.

Cause: The system cannot perform this action.

Action: Contact your system administrator or try again.

Error message: STB: 192: Cannot write file '<FILE_NAME>' because file name is not valid. Check the entered file name in the dialog box.

Cause: The specified file name is not valid.

Action: Check the spelling of file name and try again.

Error message: STB: 194: Cannot execute system command '<COMMAND>'. Look at messages of the terminal window. Please call your system administrator.

Cause: The system cannot perform this action.

Action: Contact your system administrator or try again.

Error message: STB: 196: Cannot open file '<FILE_NAME>'. Look at messages of the terminal window. Please call your system administrator.

Cause: The system cannot perform this action.

Action: Contact your system administrator or try again.

Error message: STB: 198: Cannot rename the object '<OBJ_ID>'. To save the modifications please press the Return key in the text field and try again.

Cause: The edit field was left without clicking on <**Return**> to save.

Action: Click on <**Return**> before leaving text field.

Error message: STB: 200: Cannot rename the comment '<COMMENT>'. To save the modifications please press the Return key in the text field and try again.

Cause: The edit field was left without clicking on <**Return**> to save.

Action: Click on <**Return**> before leaving text field.

Error message: STB: 202: Cannot perform this action because the report view is not set. Check report view and try again.

Cause: No report view is set.

Action: Specify a valid view and try again.

Error message: STB: 204: Cannot perform this action because the report layout format is not set. Check report layout format and try again.

Cause: No report layout format is set.

Action: Specify a valid layout format and try again.

5.1.3 Template Builder Error Messages

Error message:TEB: 100: Not implemented!

Cause: This menu function is not implemented in this release.

Action: No action is necessary. You cannot use this menu function in this release.

Error message:TEB: 101: Cannot delete object! Object is loaned.

Cause: Loan flag is set to show that object cannot be worked on by direct access tools.

Action: Try it again if loan flag is unset.

Error message:TEB: 102: Field is protected against update!

Cause: You tried to update a field that does not allow updates.

Action: No action is necessary. You cannot update this field in this template.

Error message:TEB: 103: Loaned rows were detected!

Cause: You tried to update or delete objects having a loan flag.

Action: No action is necessary. You cannot update or delete objects having a loan flag.

Error message:TEB: 104: PROC_OE_FILL_FIELD_ATT: NO_DATA_FOUND in OBJ_DICT.

Cause: The object type definition of the data base is inconsistent.

Action: Contact your data base administrator.

Error message:TEB: 105: No list of values exists!

Cause: You tried to perform the list of values function on a field which does not have a list of allowed values defined.

Action: Do not use <**List**>.

Error message:TEB: 106: No parts exist!

Cause: The object type definition of the data base is inconsistent.

Action: Contact your data base administrator.

Error message:TEB: 107: Parent object id must be entered or cancel to quit!

Cause: You tried to operate the Template Builder without a parent object.

Action: You cannot execute the Template Builder without a parent object because all objects listed are related to a specific parent object. Enter a valid parent object.

Error message:TEB: 108: PAR_OBJ_ID does not exist! Please re-enter.

Cause: You tried to operate the Template Builder without a valid parent object.

Action: You cannot execute the Template Builder without a parent object because all objects listed are related to a specific parent object. Enter a valid parent object.

Error message:TEB: 109: Control file name must be entered or cancel to quit!

Cause: You tried to perform the import function without a control file.

Action: You cannot execute the import function without a control file. Enter a valid control file.

Error message:TEB: 110: Key not allowed in this tool!

Cause: You tried to use a key or a key combination which is not allowed in the Template Builder.

Action: No action is necessary.

Error message:TEB: 111: Key not allowed in this block!

Cause: You tried to use a key or a key combination which is not allowed in this block.

Action: No action is necessary.

Error message:TEB: 112: Menu item not allowed in this block!

Cause: You tried to use a menu item which is not allowed in this block.

Action: No action is necessary.

Error message:TEB: 113: At last record!

Cause: You pressed **Next Record** while the cursor was at the last record.

Action: No action is necessary.

Error message:TEB: 114: Menu item not allowed in object type 'PROJECT'!

Cause: You tried to use a menu item which is not allowed for PROJECT object type.

Action: No action is necessary.

Error message:TEB: 115: At first record! Cannot duplicate field.

Cause: You tried to use **Copy Field** menu item with cursor at the first record.

Action: No action is necessary. You cannot execute this menu item if there is no previous record to copy the value from.

Error message:TEB: 116: Cannot perform menu item. Put the cursor into row with existing OBJ_ID!

Cause: You tried to use the **Link - Of Object** menu item with cursor at the record without an object ID.

Action: Put cursor at record with a valid object ID before you use **Link - Of Object**.

Error message:TEB: 117: Menu item not allowed in insert mode!

Cause: You tried to use **Edit - Copy** menu item on a cleared forms buffer.

Action: Put the cursor at the record with a valid object ID before you use **Edit - Copy**.

Error message:TEB: 118: Description not defined in 'OBJ_DICT'!

Cause: You tried to use **Help - On Context** menu item on a field that has no help text.

Action: Contact your data base administrator.

Error message:TEB: 119: Field must be entered!

Cause: You have not entered a value (or you have deleted a value) in a field that requires data input.

Action: You must enter a value in this field.

Error message:TEB: 120: Cannot update object! Object is loaned.

Cause: Loan flag is set to show that object cannot be worked on by direct access tools.

Action: Try it again if loan flag is unset.

Error message:TEB: 121: At last value!

Cause: You pressed **Next Record** while the cursor was at the last value.

Action: No action is necessary.

Error message:TEB: 122: At first value!

Cause: You pressed **Previous Record** while the cursor was at the first value.

Action: No action is necessary.

Error message:TEB: 123: Value not allowed. Enter Y or N!

Cause: You tried to enter a value that is not allowed.

Action: You must enter the value 'Y' or 'N'.

Error message:TEB: 124: At last attribute!

Cause: You pressed **Next Record** while the cursor was at the last attribute.

Action: No action is necessary.

Error message:TEB: 125: At first attribute!

Cause: You pressed **Previous Record** while the cursor was at the first attribute.

Action: No action is necessary.

Error message:TEB: 126: Value not allowed. Enter X or ' '!

Cause: You tried to enter a value that is not allowed.

Action: You must enter the value 'X' or ' '.

Error message:TEB: 127: Too many attributes selected! Please re-enter.

Cause: You tried to select too many attributes.

Action: You must overwrite value 'X' to ' ' from the current cursor position up to the end of all attributes.

Error message:TEB: 128: Value must be between 1 and 9!

Cause: You tried to enter values out of range 1 and 9.

Action: You must enter values between 1 and 9.

Error message:TEB: 129: Field must be unique!

Cause: You tried to enter the same value more than once.

Action: You must enter unique values.

Error message:TEB: 130: Object type is incorrect!

Cause: The object type definition of the data base is inconsistent.

Action: Contact your data base administrator.

Error message:TEB: 131: No attributes are defined within this part!

Cause: The object type definition of the data base is inconsistent.

Action: Contact your data base administrator.

Error message: **TEB: 132: At last part!**

Cause: You pressed **Next Record** while the cursor was at the last part.

Action: No action is necessary.

Error message: **TEB: 133: At first part!**

Cause: You pressed **Previous Record** while the cursor was at the first part.

Action: No action is necessary.

Error message: **TEB: 134: Value not allowed. Enter Y or N!**

Cause: You tried to enter a value that is not allowed.

Action: You must enter the value 'Y' or 'N'.

Error message: **TEB: 135: Value not allowed. Enter D (Delimited) or F (Formatted)!**

Cause: You tried to enter a value that is not allowed.

Action: You must enter the value 'D' (Delimited) or 'F' (Formatted).

Error message: **TEB: 136: Data file name must be entered or cancel to quit!**

Cause: You tried to perform the import function without a data file.

Action: You cannot execute the import function without a data file. Enter a valid data file.

Error message: **TEB: 137: Value not allowed. Enter Y (Update) or N (No update)!**

Cause: You tried to enter a value that is not allowed.

Action: You must enter the value 'Y' (Update) or 'N' (No update).

Error message: **TEB: 138: Value must be between 1 and 999!**

Cause: You tried to enter values out of range 1 to 999.

Action: You must enter values between 1 and 999.

Error message: **TEB: 139: Menu item only allowed in enter query mode!**

Cause: You tried to use **Query-Cancel** menu item while the Template Builder was not in enter query mode.

Action: No action is necessary. You can only use **Query - Cancel** in enter query mode.

Error message: **TEB: 140: Cannot toggle form. Put the cursor into row with valid OBJ_ID!**

Cause: You tried to toggle the form while the cursor was at a record with no valid object ID.

Action: Put the cursor at the record with a valid object ID if you toggle the form.

Error message: **TEB: 141: Cannot toggle form. Save the changes you have made first!**

Cause: You tried to toggle the form without saving the changes before.

Action: You must save the changes you have made first.

Error message: **TEB: 142: No valid file name! Please re-enter.**

Cause: You tried to enter an invalid file name.

Action: Enter a valid file name.

5.1.4 From AdvaBuild Structure Builder Object Information

Error message: **STB: 650: Not implemented!**

Cause: This menu function is not implemented in this release.

Action: No action is necessary. You cannot use this menu function in this release.

Error message: **STB: 651: Key not allowed in this tool!**

Cause: You tried to use a key or a key combination which is not allowed in the Object Information form.

Action: No action is necessary.

Error message: **STB: 652: Menu item not allowed in this block!**

Cause: You tried to use a menu item which is not allowed in this block.

Action: No action is necessary.

Error message: **STB: 653: Delete record not allowed in this tool!**

Cause: You tried to delete a record in this tool which is not allowed.

Action: No action is necessary.

Error message: **STB: 654: Insert record not allowed in this tool!**

Cause: You tried to insert a record in this tool.

Action: No action is necessary.

Error message: **STB: 655: At last record!**

Cause: You pressed **Next Record** while the cursor was at the last record.

Action: No action is necessary.

Error message: **STB: 656: Only T (Tree) allowed in object type ’PROJECT’!**

Cause: You tried to enter value ‘L’ into DISP field for PROJECT object type.

Action: No action is necessary.

Error message: **STB: 657: Only T (Tree) allowed because child object instance exists!**

Cause: You tried to enter value ‘L’ into DISP field for an object which has child object instances. You can only overwrite value ‘T’ to ‘L’ if no child object instances exist any more.

Action: No action is necessary.

Error message: **STB: 658: Value not allowed. Enter T (Tree) or L (List)!**

Cause: You tried to enter a value that is not allowed.

Action: You must enter the value ‘T’ (Tree) or ‘L’ (List).

Error message: **STB: 659: Key not allowed in this block!**

Cause: You tried to use a key or a key combination which is not allowed in this block.

Action: No action is necessary.

Error message: **STB: 660: Value not allowed. Enter Y or N!**

Cause: You tried to enter a value that is not allowed.

Action: You must enter the value ‘Y’ or ‘N’.

Error message: **STB: 661: At last attribute!**

Cause: You pressed **Next Record** while the cursor was at the last attribute.

Action: No action is necessary.

Error message: **STB: 662: At first attribute!**

Cause: You pressed **Previous Record** while the cursor was at the first attribute.

Action: No action is necessary.

Error message: **STB: 663: Value not allowed. Enter X or ’ ’!**

Cause: You tried to enter a value that is not allowed.

Action: You must enter the value ‘X’ or ‘ ’.

Error message: **STB: 664: Too many attributes selected! Please re-enter.**

Cause: You tried to select too many attributes.

Action: You must overwrite value ‘X’ to ‘ ‘ from the current cursor position up to the end of all attributes.

Error message: **STB: 665: Cannot print! No attributes selected. Please re-enter.**

Cause: You tried to print a file without selecting attributes.

Action: You must select at least one attribute to print a file.

Error message: **STB: 666: No valid file name! Please re-enter.**

Cause: You tried to enter an invalid file name.

Action: Enter a valid file name.

Error message: STB: 667: Value must be between 1 and 9!

Cause: You tried to enter values out of range 1 and 9.

Action: You must enter values between 1 and 9.

Error message: STB: 668: Field must be unique!

Cause: You tried to enter the same value more than once.

Action: You must enter a unique value.

Error message: STB: 669: Menu item only allowed in enter query mode!

Cause: You tried to use the **Query - Cancel** menu item while the form was not in enter query mode.

Action: No action is necessary. You can only use **Query - Cancel** in enter query mode.

5.1.5 Link Handling Messages

Error message:OML: 100: Not implemented!

Cause: This menu function is not implemented in this release.

Action: No action is necessary. You cannot use this menu function in this release.

Error message:OML: 101: Working...

Cause: Shows that the form is working for you.

Action: No action is necessary.

Error message:OML: 102: Cannot delete link! Link is exported.

Cause: Loan flag is set to show that link cannot be worked on by direct access tools.

Action: Try it again if loan flag is unset.

Error message:OML: 103: Exported rows were detected!

Cause: You tried to update or delete links having a loan flag.

Action: No action is necessary. You cannot update or delete links having a loan flag.

Error message:OML: 104: No link types exist!

Cause: The link type definition of the data base is inconsistent.

Action: Contact your data base administrator.

Error message:OML: 105: PAR_OBJ_ID does not exist! Please re-enter.

Cause: You tried to operate the Link Handling without a valid parent object.

Action: Enter a valid parent object or use the Link Handling without a parent object.

Error message:OML: 106: Cannot select objects!

Cause: You tried to select objects via the Query Condition Window dialog box.

Action: Edit the values of the Query Condition Window dialog box to get objects.

Error message: **OML: 107: It takes a while! Objectlist has more then 1000 objects.**

Cause: You tried to import an objectlist has more then 1000 objects.

Action: No action is necessary. It takes a while until all objects are imported.

Error message: **OML: 108: Cannot select object! Object type not allowed for this link type.**

Cause: Communication between Structure Builder and Link Handling is incorrect.

Action: Contact your data base administrator.

Error message: **OML: 109: Cannot update link! Link is exported.**

Cause: Loan flag is set to show that link cannot be worked on by direct access tools.

Action: Try it again if loan flag is unset.

Error message: **OML: 110: Key not allowed in this block!**

Cause: You tried to use a key or a key combination which is not allowed in this block.

Action: No action is necessary.

Error message: **OML: 113: Key not allowed in this tool!**

Cause: You tried to use a key or a key combination which is not allowed for Link handling.

Action: No action is necessary.

Error message: **OML: 114: Link handling tool started without query!**

Cause: Communication between Structure Builder and Link Handling is incorrect.

Action: Contact your data base administrator.

Error message: **OML: 115: Menu item not allowed in this block!**

Cause: You tried to use a menu item which is not allowed in this block.

Action: No action is necessary.

Error message: **OML: 116: At first record! Cannot duplicate field.**

Cause: You tried to use **Copy Field** menu item with cursor at the first record.

Action: No action is necessary. You cannot execute this menu item if there is no previous record to copy the value from.

Error message: **OML: 117: Menu item not allowed in insert mode!**

Cause: You tried to use **Edit - Copy** menu item on a cleared forms buffer.

Action: Put the cursor at the record with a valid object ID before you use **Edit - Copy**.

Error message: **OML: 118: Cannot perform menu item. Put the cursor into OBJ_ID1 or OBJ_ID2!**

Cause: You tried to use **Object - Open** or **Link - Open Type** menu items while the cursor was out of the OBJ_ID1 or OBJ_ID2 fields.

Action: Put the cursor in either the OBJ_ID1 or OBJ_ID2 field before using menu **Object - Open** or **Link - Open Type**.

Error message: **OML: 120: Only one object can be open. Enter N!**

Cause: You can only search for close links and open links. Open links means that only one side (left or right) can be opened.

Action: No action is necessary.

Error message: **OML: 121: Value not allowed. Enter Y or N!**

Cause: You tried to enter a value that is not allowed.

Action: You must enter the value 'Y' or 'N'.

Error message: **OML: 122: At last link type!**

Cause: You pressed **Next Record with** the cursor at the last link type.

Action: No action is necessary.

Error message: **OML: 123: Value must be between 1 and 999!**

Cause: You tried to enter values out of range 1 to 999.

Action: You must enter values between 1 and 999.

Error message: **OML: 125: PROC_LH_FILL_FIELD_ATT: NO_DATA_FOUND in LINK_DICT.**

Cause: The link type definition of the data base is inconsistent.

Action: Contact your data base administrator.

Error message: **OML: 126: No list of values exists!**

Cause: You tried to perform the list of values function on a field which does not have a list of allowed values defined.

Action: Do not use <**List**>.

Error message: **OML: 127: Description not defined in 'LINK_DICT'!**

Cause: You tried to use **Help - On Context** menu item on a field that has no help text.

Action: Contact your data base administrator.

Error message: **OML: 128: Field is protected against update!**

Cause: You tried to update a field that does not allow updates.

Action: No action is necessary. You cannot update this field.

Error message: **OML: 129: At last value!**

Cause: You pressed **Next Record** while the cursor was at the last value.

Action: No action is necessary.

Error message: **OML: 130: At first value!**

Cause: You pressed **Previous Record** while the cursor was at the first value.

Action: No action is necessary.

Error message: **OML: 131: Linktype is incorrect!**

Cause: The link type definition of the data base is inconsistent.

Action: Contact your data base administrator.

Error message: **OML: 132: No attributes are defined within this part!**

Cause: The link type definition of the data base is inconsistent.

Action: Contact your data base administrator.

Error message: **OML: 133: At last part!**

Cause: You pressed **Next Record** while the cursor was at the last part.

Action: No action is necessary.

Error message: **OML: 134: At first part!**

Cause: You pressed **Previous Record** while the cursor was at the first part.

Action: No action is necessary.

Error message: **OML: 135: No parts exist!**

Cause: The link type definition of the data base is inconsistent.

Action: Contact your data base administrator.

Error message: **OML: 136: Field must be entered!**

Cause: You have not entered a value (or you have deleted a value) in a field that requires data input.

Action: You must enter a value in this field.

Error message: **OML: 139: At last attribute!**

Cause: You pressed **Next Record** while the cursor was at the last attribute.

Action: No action is necessary.

Error message: **OML: 140: At first attribute!**

Cause: You pressed **Previous Record** while the cursor was at the first attribute.

Action: No action is necessary.

Error message: **OML: 141: Value not allowed. Enter X or ’ ’!**

Cause: You tried to enter a value that is not allowed.

Action: You must enter the value ‘X’ or ‘ ’.

Error message: **OML: 142: Too many attributes selected! Please re-enter.**

Cause: You tried to select too many attributes.

Action: You must overwrite value ‘X’ to ‘ ‘ from the current cursor position up to the end of all attributes.

Error message: **OML: 143: Cannot print! No attributes selected. Please re-enter.**

Cause: You tried to print a file without selecting attributes.

Action: You must select at least one attribute to print a file.

Error message: **OML: 144: No valid file name! Please re-enter.**

Cause: You tried to enter an invalid file name.

Action: Enter a valid file name.

Error message: **OML: 145: Cannot perform menu item. Put the cursor into row with existing OBJ_ID!**

Cause: You tried to use **Object - Open** or **Link - Open Type** menu items while the cursor was at the record without object ID’s.

Action: Put the cursor at the record with a valid object ID’s before you use **Object - Open or Link - Open Type**.

Error message: **OML: 146: Value must be between 1 and 9!**

Cause: You tried to enter values out of range 1 and 9.

Action: You must enter values between 1 and 9.

Error message: **OML: 147: Field must be unique!**

Cause: You tried to enter the same value more than once.

Action: You must enter a unique value.

Error message: **OML: 148: Cannot perform menu item. No query defined!**

Cause: Communication between Structure Builder and Link Handling is incorrect.

Action: Contact your data base administrator.

Error message: **OML: 149: At last record!**

Cause: You pressed **Next Record** while the cursor was at the last record.

Action: No action is necessary.

Error message: **OML: 150: Menu item only allowed in enter query mode!**

Cause: You tried to use **Query - Cancel** menu item while the Link Handling was not in enter query mode.

Action: No action is necessary. You can only use **Query - Cancel** in enter query mode.

Error message: **OML: 151: Expression too long! Only 234 character allowed. Please re-enter.**

Cause: You tried to enter an additional condition which is longer than 234 characters.

Action: Write your additional condition not longer than 234 characters maximum.

Error message: **OML: 152: Cannot enter a query! All fields are predefined in 'Set New View Directed'.**

Cause: You tried to start the enter query mode while the LINK_TYPE, OBJ_ID1, OBJ_TYPE1, OBJ_ID2, OBJ_TYPE2 fields are already predefined via the Set New View Directed dialog box. You can only query those attributes that are not already defined via the Set New View Directed dialog box.

Action: No action is necessary.

Error message: **OML: 153: Enter query mode not allowed in 'Set New View Undirected'!**

Cause: You tried to start the enter query mode while the last execution was Set New View Undirected dialog box (Link Of Object). If the Set New View Undirected dialog box is executed last enter query mode is not available.

Action: You have to execute Set New View Directed dialog box last in order to get the enter query mode.

Error message: **OML: 154: Import objectlist finished!**

Cause: Shows that the import of objectlist is finished.

Action: No action is necessary.

Error message: **OML: 155: Import objectlist only for different users at the same time allowed!**

Cause: You tried to start the import of objectlist in a second form without saving the imported objects in the first form. Note that only different users can start this function at the same time.

Action: Save your imported objects in the first form first before you import the next objectlist in the second form.

Error message: **OML: 156: Cannot import objectlist! No objects defined.**

Cause: You tried to start the import of objectlist when the objectlist has no objects.

Action: Create an objectlist first which has object ID's before you start the import.

Error message: **OML: 157: Importing objects finished!**

Cause: Shows that the import of the selected objects is finished.

Action: No action is necessary.

5.1.6 Plant Documentation Builder Error Messages

Error message: **DOC: 1 : AdvaBuild Plant Documentation Builder - Configure Version 2.1/0!**

Cause: Answer for **Help on Version** Menu Item**.**

Action:

Error message: **DOC: 9 : Key not allowed in this block!**

Cause: This function or key is not implemented or not allowed in this block (Header- or Order-Block).

Action: Refer to Section 1.9, User Interface for AdvaBuild Basic Software for available keys.

Error message: **DOC: 13 : Value not allowed. Please enter 'X' or 'Blank'!**

Cause: The input for the Select column for print orders is not valid.

Action: Replace the actual input and choose "X" to activate the print order or "blank" to suppress.

Error message: **DOC: 16 : Not a valid file name! Please re-enter!**

Cause:

Action:

Error message: **DOC: 108 : Function not implemented!**

Cause: This function is not implemented or not applicable in this context.

Action: Refer to applicable section in this book.

Error message: **DOC: 109 : Function not allowed in Query-Mode!**

Cause: The Input Mode functions can not be used in the Query Mode.

Action: Quit the Query Mode and repeat your input.

Error message: **DOC: 110 : Function not allowed in this context!**

Cause: The desired function is not allowed in this context.

Action: Check the User Manual and/or your command.

Error message: **DOC: 111 : Function not allowed in this mode!**

Cause: Same as 110 for mode.

Action:

Error message: **DOC: 112 : You are not in the Query-Mode!**

Cause: Queries are not possible in the Input Mode.

Action: To enter a query, enter the Query Mode, select your query and execute.

Error message: **DOC: 113 : No record to duplicate or function not allowed!**

Cause: The range of commands is not allowed (not possible) in this context.

Action: Check the User Manual (especially the part Menu / Forms) and/or your command.

Error message: **DOC: 114 : Menu item not allowed in this block!**

Cause: Different usage of Menu Items in Header- or Order Block.

Action: Check the User Manual.

Error message: **DOC: 115 : Menu item not allowed in Insert-Mode!**

Cause: Queries are not possible in the Input Mode.

Action: To enter a query, enter the Query Mode, select your query and execute.

Error message: **DOC: 116 : Invalid value for 'OBJ_TYPE' or 'COM_VIEW'!**

Cause: The key combination OBJ_TYPE + COMMAND + COM_VIEW is not a Print Command available in the table OBJ_COMMAND.

Action: Use the function List_of_Values to choose an available Print Command via combination OBJ_TYPE + COMMAND + COM_VIEW.

Error message: **DOC: 117 : Press <List> to see possible values!**

Cause: Next line (Action) of message DOC 116.

Action: -

Error message: **DOC: 118 : Invalid value, ’A’, ’N’ or ’H’ expected!**

Cause: Invalid input for Activation flag.

Action: Select “A” to Activate a message or
“N” to set a message “Not Active” or
“H” to set a message to Hibernating state.

Error message: **DOC: 119 : Invalid value, ’Y’ or ’N’ expected!**

Cause: Invalid input for “Additional Attribute exist”.

Action: Check the User Manual; Only ’Y’ or ’N’ allowed.

Error message: **DOC: 120 : Invalid value, ’D3L’, ’I3L’, ’I3P’, ’I4L’ or ’I4P’ expected!**

Cause: Invalid Input.

Action: Check the User Manual.

Error message: **DOC: 121 : Invalid value, ’A4L’, ’A4P’, ’Q4L’ or ’Q4P’ expected!**

Cause: Invalid Input for “Format”.

Action: Check the User Manual

Error message: **DOC: 122 : Invalid value, ’I’ or ’E’ expected!**

Cause: Invalid Input for “Subtree Handling”.

Action: Check the User Manual

Error message: **DOC: 123 : Value must be between 1 and 999!**

Cause: Invalid Input for Cycle Time or Number of cycles.

Action: Check the User Manual. The waiting time for printouts of calls to Builders is “Number of cycles” times wait the Cycle Time.

Error message: **DOC: 124 : Could not read internal parameter file (c_llc_int.dat)!**

Cause: Internal system error.

Action: Contact your system manager.

Error message: **DOC: 125 : PDB - Please use the corresponding STB Menu Item!**

Cause: Function is only implemented on STB level.

Action:

Error message: **DOC: 126 : Enter application and subject and execute with MENU/QUERY/EXECUTE!**

Cause: Operating instruction only - The next step in range.

Action:

Error message: **DOC: 1201 : PDB : Package Printout is started!**

Cause: Execution message only.

Action: Wait for end of Printout procedure.

Error message: **DOC: 1202 : PDB : Package Generation finished!**

Cause: Execution message only. The Package Generation is finished.

Action: Repeat or leave the Package Generation utility.

Error message: **DOC: 1203 : Press Accept/Commit to print or Exit/Cancel to abort action!**

Cause: Operating message about the next step.

Action:

Error message: **DOC: 1204 : Please enter value for : Status**

Cause: Operating message about the actual step.

Action:

Error message: **DOC: 1205 : Please enter the new Revision Index**

Cause: Operating message about the actual step.

Action:

Error message: **DOC: 1206 : Please enter the Revision notes**

Cause: Operating message about the actual step.

Action:

Error message: **DOC: 1207 : Please enter your name (sign)**

Cause: Operating message about the actual step.

Action:

Error message: **DOC: 1208 : Saves Revision Changes and starts generation /Press 'Accept/Commit' if ready.**

Cause: Operating message about the next step.

Action:

Error message: **DOC: 1209 : Cancels Revision Changes and starts generation /Press 'Accept/Commit' if ready.**

Cause: Operating message about the next step.

Action:

Error message: **DOC: 1210 : Help to Revision Window / Press ’Accept/Commit’ if ready!**

Cause: Operating message about the actual step.

Action:

Error message: **DOC: 1211 : Change statusflag if necessary (F=Fail,S=Succeeded,X=Generate)**

Cause: Operating message about the actual step.

Action:

Error message: **DOC: 1212 : Please enter value for : DOC_TYPE**

Cause: Operating message about the actual step.

Action:

Error message: **DOC: 1213 : Please enter value for : FILENAME**

Cause: Operating message about the actual step.

Action:

Error message: **DOC: 1214 : Enter ’X’ or ’ ’. Press ’Return’ to continue, ’Exit’ to cancel**

Cause: Operating message about the actual step.

Action:

Error message: **DOC: 1215 : Enter filename. Press ’Return’ to continue, ’Exit’ to cancel**

Cause: Operating message about the actual step.

Action:

Error message: **DOC: 1216 : Executes the print / Press ’Accept/Save’ if ready!**

Cause: Operating message about the actual step.

Action:

Error message: **DOC: 1217 : Cancels the print / Press ’Accept/Save’ if ready!**

Cause: Operating message about the actual step.

Action:

Error message: **DOC: 1218 : Help not yet implemented / Press ’Accept/Save’ if ready**

Cause:

Action:

5.1.7 Export/Import Error Messages

Error message: **EIU: 1: Cannot fork to start a sub-process. Possibly operation system problem.
Please consider the next message and contact your system administrator.**

Cause: An attempt to start a sub-process failed. A further operation system message may provide information on the reason.

Action: Read the following operation system message. If you cannot correct the problem contact your system administrator.

Error message: **EIU: 2: Cannot connect to database as '<USER_ID>'/'<PASS_WORD>'.
Please contact your database administrator.**

Cause: Connect to database failed.

Action: Check the spelling of your database login. If it is correct, contact your database administrator for further help.

Error message: **EIU: 3: Cannot find import data file or control file specification in the command line. Parameter is missing.
Please check your call to eiu.**

Cause: The call to eiu is not correct: there is no import data or control file specified.

Action: Check syntax and spelling of the command line.

Error message: **EIU: 4: Cannot find export data file or select file specification in the command line. Parameter is missing.
Please check your call to eiu.**

Cause: The call to eiu is not correct: there is no export data or select file specified.

Action: Check syntax and spelling of the command line.

Error message: **EIU: 5: Cannot find the environment variable C_PROJ_EIU.
Please specify this variable.**

Cause: The environment variable C_PROJ_EIU, which specifies the directory for input and output files, is not found.

Action: Specify the environment variable C_PROJ_EIU.

Error message: **EIU: 6: Cannot find first command line attribute 'import' or 'export'.
Please specify this parameter.**

Cause: The first command line parameter *import* or *export* is missing.

Action: Check syntax and spelling of the command line.

Error message: **EIU: 7: Cannot allocate memory. No system memory available.
Please contact your system administrator.**

Cause: An attempt to allocate memory by the standard operating system memory allocation functions failed. Maybe there is no system memory available.

Action: Try again. In case of further failure, ask your system administrator for advice.

Error message: **EIU: 8: Cannot identify command line parameter ’<PARAMETER>’.
Please check syntax and spelling.**

Cause: There is an unknown command line parameter.

Action: Check syntax and spelling of the command line.

Error message: **EIU: 9: Do not know which extension to attach to ’<FILE>’ file filename ’<PARAMETER>’.
Please specify filenames with extensions.**

Cause: A file name was specified without an extension.

Action: Use filenames with extensions only.

Error message: **EIU: 10: Cannot find the command line parameter ’<PARAMETER>’.
Please specify ’<PARAMETER>’.**

Cause: A necessary command line parameter is missing.

Action: Specify the required parameter.

Error message: **EIU: 11: Cannot find type of object or link to import. Import needs one of the command line parameters ’sel’, ’oty’ or ’lty’.
Please specify one of these parameters.**

Cause: An import needs either a command line *oty* or *lty* definition or the *sel* parameter for a select file definition. None of those parameters was specified.

Action: Specify one of those parameters.

Error message: **EIU: 12: Cannot open log file. There might be an operation system problem.
Please consider the next message and contact your system administrator.**

Cause: The log file could not be opened. A further operation system message may provide information on the reason.

Action: Read the following operation system message. If you cannot fix the problem please contact your system administrator.

Error message: **EIU: 100: Control file: Table name missing.**

Cause: No table name has been specified in the control file.

Action: Check sytax and spelling of the control file and specify the name of the temporary table in the control file.

Error message: **EIU: 101: Control file: Attribute names missing (’(’ not found).**

Cause: The attribute names of the temporary table are specified in the control file within brackets. These brackets cannot be found.

Action: Check syntax and spelling of the control file and make sure the attribute specifications are imported.

Error message: **EIU: 102: Attribute ‘<ATT_NAME>’ may not be accessed (internal attribute).**

Cause: The AdvaBuild database contains internal attributes, which may not be changed by the user. There has been an attempt to change the value of such an attribute.

Action: Change your input files to make sure no internal attributes are changed.

Error message: **EIU: 103: Control file: Attribute type or length missing for attribute ‘<ATT_NAME>’.**

Cause: In contrast to the SQL*Loader description type *and* length specifications are compulsory for the control file, because they are used to create the temporary import table. Such a specification is missing for the specified attribute.

Action: Please check the control file and make sure, that all attribute types and corresponding length fields are specified.

Error message: **EIU: 104: Control file: Next attribute missing (’)’ not found).**

Cause: The attribute specification in the control file is enclosed in brackets. The closing bracket was not found.

Action: Add the missing bracket or correct the attribute specifications.

Error message: **EIU: 105: Control file: Unexpected end of file.**

Cause: The control file ends unexpectedly.

Action: Check the syntax of your control file.

Error message: **EIU: 106:Parent object ID ‘<P_OBJECT>’ does not exist.**

Cause: To import an object a parent has to be specified. The parent which has been provided does not exist.

Action: Check the parent specification.

Error message: **EIU: 107: Link exists but cannot be read (database consistency problem).**

Cause: A link that could formerly be selected cannot be selected any more. The reason may be an internal error or an inconsistent database.

Action: Try again to import the link. In case of repeated failure contact your database administrator for database consistency checks.

Error message: **EIU: 108: Control file: Missing length or digit-number of datatype VARCHAR or DECIMAL.**

Cause: The datatypes VARCHAR and DECIMAL need length or digit-number specifications resp. Such a specification is missing in the control file.

Action: Check syntax and spelling of your control file and correct it.

Error message: **EIU: 109: Control file: Datatypes SMALLINT and DOUBLE should not have length specifications.**

Cause: The datatypes SMALLINT and DOUBLE may not have any length specifications in the select file, but such a specification was found.

Action: Check syntax and spelling of your control file and correct it.

Error message: **EIU: 110: Control file: Quotation mark missing.**

Cause: A quotation mark within the control file is missing.

Action: Check syntax and spelling of your control file and correct it.

Error message: **EIU: 112: Select file: Keyword CATEGORY missing.**

Cause: The keyword CATEGORY, which always starts a new select block within the select file, is missing.

Action: Check the syntax and spelling of the select file and supplement this keyword.

Error message: **EIU: 113: Select file: Keyword TASK missing.**

Cause: The select file keyword TASK, which specifies the task to perform is missing.

Action: Check the syntax and spelling of the select file and supplement this keyword.

Error message: **EIU: 114: Select file: Reading of the category failed.**

Cause: The value for the category in the select file is wrong (allowed values are *objects* or *links*).

Action: Check the spelling of this value (allowed values are *objects* and *links*).

Error message: **EIU: 115: Select file: Keyword SELECT missing.**

Cause: The select file keyword SELECT is missing.

Action: Check the syntax and spelling of the select file and supplement this keyword.

Error message: **EIU: 116: Select file: Reading of the task failed.**

Cause: The value for the task in the select file is wrong.

Action: Check the spelling of this value (allowed values are *insert*, *update* or *close*).

Error message: **EIU: 117: Select file: PARENT not an allowed keyword for links.**

Cause: The select file parameter PARENT, which may only be used for category objects, is used for category links.

Action: Remove this parameter from select blocks for links.

Error message: **EIU: 118: Select file: Keyword ATTRIBUTE missing.**

Cause: The attribute specification of a select block is missing.

Action: Check syntax and spelling of the select file.

Error message: **EIU: 119: Select file: Keyword FROM missing.**

Cause: The select file keyword FROM, required to specify the tables to select data from, is missing.

Action: Check syntax and spelling of the select file and supplement this keyword.

Error message: **EIU: 120: Select file: Keyword WHERE missing.**

Cause: The select file keyword WHERE, necessary to specify a where condition for a select of data, is missing. This keyword CAN NOT be ommitted, even if no where condition follows.

Action: Check syntax and spelling of the select file and supplement this keyword.

Error message: **EIU: 121: Select file: Reading of the group failed.**

Cause: The value specified for the select file keyword GROUP is not correct.

Action: Check the spelling of this value (allowed values are *obj*, *sta* and *lnk*).

Error message: **EIU: 122: Internal error: Cannot find attribute name in list (memory problem)!**

Cause: Internal memory problem (memory has been destroyed).

Action: Try again. Check your system memory. In case of repeated failure contact ABB.

Error message: **EIU: 123: Database error: Table entry was deleted, access failed!**

Cause: An entry to an AdvaBuild database, which was inserted shortly before, has been deleted. Another user or tool has destroyed the database.

Action: Contact your database administrator for help.

Error message: **EIU: 124: Extract file: File object '<OBJ_ID>' does not exist.**

Cause: An object from which a file is to be extracted does not exist

Action: Please check the object ID specifications in your data file.

Error message: **EIU: 200: Error occurred while calling LLC or operation system functions.**

Cause: Another tool has caused an error. General message to sum up EIU errors caused by library or database functions.

Action: General message. Check the log files for the error, which caused this message.

5.1.8 From AdvaBuild On-line Builder

Error message: **ONB: 9: Cannot find a file or no right for attempted file access.**

Cause: A PU command from $C_BIN_ONF_BIN has no access to create, read or write a specific file.

Action: Check the access right for the associated UXBase/ and onf/ directories.

Error message: **ONB: 11: Cannot execute PU command.**

Cause: A PU command from $C_BIN_ONF_BIN was called with wrong parameter or parameter out of range.

Action: Internal error. Try it again.

Error message: **ONB: 12: Permission denied for attempted command '$C_BIN_ONF_BIN/PUvoldef'.
Please boot the RealTimeAccelerator board.**

Cause: The command $C_BIN_ONF_BIN/PUvoldef has no access to create a SRCE or USER volume, because there is a wrong environment.

Action: Please boot the RealTimeAccelerator board to get a defined environment.

Error message: **ONB: 13: User aborts PU command.**

Cause: A PU command from $C_BIN_ONF_BIN was aborted by the user.

Action: Try it again without aborting.

Error message: **ONB: 14: Cannot write the volume setup.
Please check the directories SRCE/ and USER/
of the selected node.**

Cause: The command $C_BIN_ONF_BIN/PUvoldef has no access to create a SRCE or USER volume.

Action: Check the access right for the associated SRCE/ and USER/ directories.

Error message: **ONB: 15: The target system is not available.**

Cause: For the given Net,Node number is no target (e.g. MP200) connected to the MB300.

Action: Use a Net,Node number of a connected target. Does the target exist but the message comes, then check the configuration of your RealTimeAccelerator board.

Error message: **ONB: 101: Cannot finish the command '$C_BIN_ONF_BIN/PUterm'.
The command terminated by signal.**

Cause: The command $C_BIN_ONF_BIN/PUterm terminated with a UNIX signal.

Action: Internal error. Try it again.

Error message: **ONB: 102: Cannot finish the command '$C_BIN_ONF_BIN/PUterm'. The command stopped by signal.**

Cause: The command $C_BIN_ONF_BIN/PUterm stopped with a UNIX signal.

Action: Internal error. Try it again.

Error message: **ONB: 103: Cannot execute the command '$C_BIN_ONF_BIN/PUterm'.**

Cause: The command $C_BIN_ONF_BIN/PUterm has no execute access.

Action: Internal error. Check the access rights of $C_BIN_ONF_BIN/PUterm.

Error message: **ONB: 104: Cannot compress or decompress the file '<FILENAME>'. Required extensions are TIX, TBX, TI or TB.**

Cause: The convert dialog tried to compress or decompress a file with a not allowed extension.

Action: Check or rename the file '<FILENAME>'.

Error message: **ONB: 105: Cannot find '01' or wrong extension in filename '<FILENAME>'.**

Cause: The convert dialog tried to convert the file '<FILENAME>'. The first file of the convert action must look like 'XXXX0501.YYY'. Allowed extensions in the PCDATA/ directory are AAX, ABX, AEX and TIX, in DBDATA/ are BAX, BBX, BEX and TBX, in SRCE/ are AA, AB, AE, BA, BB and BE, in USER/ are TI and TB.

Action: Check or rename the file '<FILENAME>'.

Error message: **ONB: 106: Cannot split the filename**
'<PATH>'.
The extension '<EXTENSION>' is not allowed here.
Required extensions are AAX, ABX, AEX, BAX, BBX or BEX.

Cause: The convert dialog tried to split a big source code file into more little files. But it found a not allowed file extension.

Action: Check or rename the file '<PATH>'.

Error message: **ONB: 107: (*NEXT FILE='<FILENAME>'*)**
Cannot find the file '<PATH>'.

Cause: The next file reference at the end of a source code contains a filename, which cannot be open at the directorypath '<PATH>'.

Action: Check the reference entry and availability of the file '<PATH>'. Normaly the referencefiles are counted.

Error message: **ONB: 108: Cannot read next filename from the line**
'<LINE>'.

Cause: The convert dialog read a line with a next file reference. But there was no allowed filename found.

Action: Check the syntax of the next file reference. The syntax must be:
(*NF=SRCE:*filename***)**

Error message: **ONB: 110: Cannot lock the RealTimeAccelerator board.
The board is already locked from another user.**

Cause: Another user is working with the RealTimeAccelerator board.

Action: Check if somebody else is just working with the On-line Builder. If yes, then you have to wait until he has finished working with the On-line Builder. If no, then is the environment for the board not correct and it must be booted.

Error message: **ONB: 115: Cannot find the file
'<PATH>'.**

Cause: The On-line Builder searched for the file which is shown in '<PATH>'. But the file was not found.

Action: Check the access rights of the searched file and of the directories.

Error message: **ONB: 117: Cannot open the file
'<PATH>'.**

Cause: The On-line Builder tried to open the file which is shown in '<PATH>'. But the file cannot be open.

Action: Check the access rights of the searched file and of the directories.

Error message: **ONB: 120: Cannot find the directory
'<PATH>'.
Please contact your system administator.**

Cause: The On-line Builder searched for the directory which is shown in '<PATH>'. But the directory was not found.

Action: Please contact your system administator. The directory must be created automaticly from the AS 500ES system.

Error message: **ONB: 122: Cannot execute UNIX command
'<COMMAND>'.**

Cause: The On-line Builder tried to execute a UNIX command on shell level.

Action: Please contact your system administator. Execute the command '<COMMAND>' on your hpterm window and see the UNIX error.

Error message: **ONB: 125: On your system is no RealTimeAccelerator board installed.**

Cause: The On-line Builder must have an installed and configurated RealTimeAccelerator board for working.

Action: Contact ABB.

Error message: **ONB: 127: Cannot start On-line Builder session because your RealTimeAccelerator board is not booted. Please boot the board with $C_BIN_ONF_ETC/rta_boot.**

Cause: After the board configuration and after a boot of the workstation the RealTimeAccelerator must be booted.

Action: Please boot the board with $C_BIN_ONF_ETC/rta_boot.

Error message: **ONB: 130: Cannot find environment variable ’<ENV_VAR>’. Please contact your system administator.**

Cause: The AS 500ES specific environment variable ’<ENV_VAR>’ was not found. They will be set during startup of the AS 500ES database.

Action: Check whether the AS 500ES database is running. If not, then start it. If yes, then check your system environment.

Error message: **ONB: 132: Cannot create date and time. System error. Check environment variable ’TZ’.**

Cause: To get the system time, the environment variable ‘TZ’ must be set.

Action: Please contact your system administator.

Error message: **ONB: 135: AdvaBuild On-line Builder software not installed completely. Please contact your system administator.**

Cause: The PU commands in the directory $C_BIN_ONF_BIN are not completely installed for the On-line Builder.

Action: Please contact your system administator. The On-line Builder must be installed new.

Error message: **ONB: 137: Cannot execute PU command.**

Cause: A PU command from $C_BIN_ONF_BIN was called with wrong parameter or parameter out of range.

Action: Internal error. Try it again.

Error message: **ONB: 145: Cannot read the content of the file ’<PATH>’. Allowed values like 001, 01, 1, 046, 46, 049 or 49.**

Cause: The On-line Builder read the language file c_language.dat of the selected node but the content is not correct.

Action: Check the content of the language file ’<PATH>’. If it has not one of the allowed value, then remove the file and start the On-line Builder again.

Error message: **ONB: 150: AdvaBuild On-line Builder not implemented for PCPGM under an object type AC110C.**

Cause: The On-line Builder was started from an object PCPGM which have the parent object AC110C. But for the AC110C object type is no On-line Builder running.

Action: Start the On-line Builder not on an object PCPGM under a AC110C, only under a MP200 object type.

Error message: **ONB: 151: Extract PC source code is not usable for PCPGM under an object type AC110C.**

Cause: Same cause as in error message ONB: 150.

Action: Start the Extract PC source code not on an object PCPGM under a AC110C, only under a MP200 object type.

Error message: **ONB: 211: Cannot find the start item designation. Enter a value in Item Designation Start PC or change the existing value.**

Cause: The textfield Start PC in the border Item Designation has no entry or an entry was edited which PC number is not available in the PC program.

Action: Enter a value in Item Designation Start PC or change the existing value.

Error message: **ONB: 212: Cannot find the stop item designation. Enter a value in Item Designation Stop PC or change the existing value. The value must end with a number or an asterisk like 8.1.1 or 8.1.***

Cause: The textfield Stop PC in the border Item Designation has no entry or an entry was edited which PC number is not available in the PC program.

Action: Enter a value in Item Designation Stop PC or change the existing value.

Error message: **ONB: 213: Cannot find the start and stop item designation. Enter a value in Item Designation Start PC and Stop PC.**

Cause: The textfields Start PC and Stop PC in the border Item Designation have no entries.

Action: Enter a value in Item Designation Start PC and Stop PC.

Error message: **ONB: 214: Cannot find the output filename. Select a filename in the list ABB Master Representation or edit the Output File Name.**

Cause: The textfield File Name in the border Output has no entry.

Action: Select a filename in the list ABB Master Representationor edit the Output File Name.

Error message: **ONB: 215: Cannot find the input filename. Select a filename in the list General Representation.**

Cause: No filename in the list of General Representation was selected.

Action: Select a filename in the list General Representation.

Error message: **ONB: 216: Place of the wildcard '*' only after the last point in the stop item designation.**
Enter a value like 8.2.*

Cause: An asterisk '*' was used as wildcard for the item designation stop PC, but not with the correct syntax.

Action: Use the wildcard only after the last point in the stop item designation.
Enter a value like 8.2.*

Error message: **ONB: 217: Cannot find a path for the extract input filename.**

Cause: The extract session searched the input filename in the PCDATA directory of the selected NODE, but the whole path does not exist or has an error.

Action: Check the directory structure of the selected NODE.

Error message: **ONB: 218: Cannot open the temporary extract output file.**

Cause: The extract session writes a temporary output file before it writes the real output file.

Action: Check the directory structure of the selected NODE and the protections of the PCDATA directory.

Error message: **ONB: 219: (*NEXT FILE=file*): Cannot find the file.**

Cause: The next file reference at the end of a source code contains a filename, which cannot be open.

Action: Check the reference entry and availability of the file in the selected NODE structure. Normaly the referencefiles are counted.

5.2 Export Import Utility Restrictions

Compared to the original control file syntax there are some restrictions:

- SQL*Loader OPTIONS should not be specified in the control file (EIU specifies all necessary options when calling the SQL*Loader)
- CONTINUE_LOAD and READBUFFERS are not allowed (as no direct path loads are performed)
- No INFILE/INDDN, BADFILE/BADDN or DISCARDFILE/DISCARDDN may be specified (INFILE is provided as command line or function parameter, the BADFILE is created by the control file name with extensions .bad, a DISCARDFILE is not needed as WHEN clauses are not allowed)
- DISCARDS or DISCARDMAX is not needed as WHEN clauses are not allowed
- An OS-dependent file processing specification string should not be defined
- The INSERT, APPEND or REPLACE clause is not needed, as the temporary data table is always a new and empty table

- The PRESERVE BLANKS clause should not be used, as the EIU always removes leading and trailing blanks from import data
- BEGINDATA is not allowed; data has to be contained in a separate data file
- The SORTED clause is not allowed and using it does not make sense, as the temporary table has no indexes
- The REENABLE clause is not allowed and not needed, as the temporary table does not contain any constraints
- WHEN-conditions are not allowed (you can use WHERE-conditions in the select file instead)
- The column specifications SYSDATE, CONSTANT and SEQUENCE are not allowed (SYSDATE and constants can be specified in the select file instead; instead of SEQUENCES use a simple RECNUM and calculate with this number in the select file)
- NULLIF, DEFAULTIF and sql_strings are not allowed
- Each control file may only describe the import of data into one table (this means there is only one INTO TABLE statement in one control file), which is only a temporary table and is removed at the end of an import
- The datatypes RAW, GRAPHIC and VARGRAPHIC cannot be properly inserted in ORACLE databases and are therefore forbidden
- The datatype VARCHAR needs an obligatory length field, which is interpreted as maximal length of a character string
- Comments, introduced by a double dash (--) have to start at the beginning of a line. If there are any characters between the beginning of a new line and the double dash introducing a comment, this might confuse the EIU
- PIECED is only used for direct path loads.

Appendix A AS500ES Directory Structure

A.1 Top Directory, Principle Scheme

The files of the AS500ES products, the engineering projects data and AS500ES user directories are separated from the rest of the HP-UX system directories, Figure A-1. The top directories are called "products", "proj" and "users" located on the root directory of the workstation.
The "products" directory comprised all the components (executable, libraries, setup files and so on) which belongs to the AS500ES product; the "proj" directory covers all the project data which are produced at project engineering activities. Below the "users" directory there are predefined directories which are named with user log-in names for the AS500ES. Most of the directories defined here are connected to an environment variable so that the user may use these to set a path (for the list of environment variables, see Section A.7, AS500ES Environment Variables).

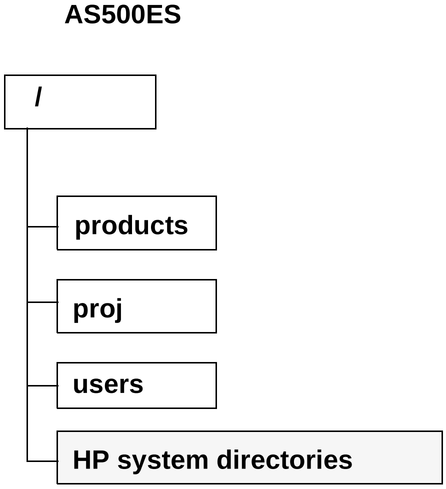


Figure A-1. AS500ES Top Directory Scheme

NOTE

Do not manipulate the directories or files. It may have tremendous impact the usability of the product.

A.2 Products Structure

Third party products like Framemaker and ORACLE as well as the executable and support files for the engineering station are stored here as shown in Figure A-2. There are also configuration files to setup the AS500ES software environment. See Figure A-3.

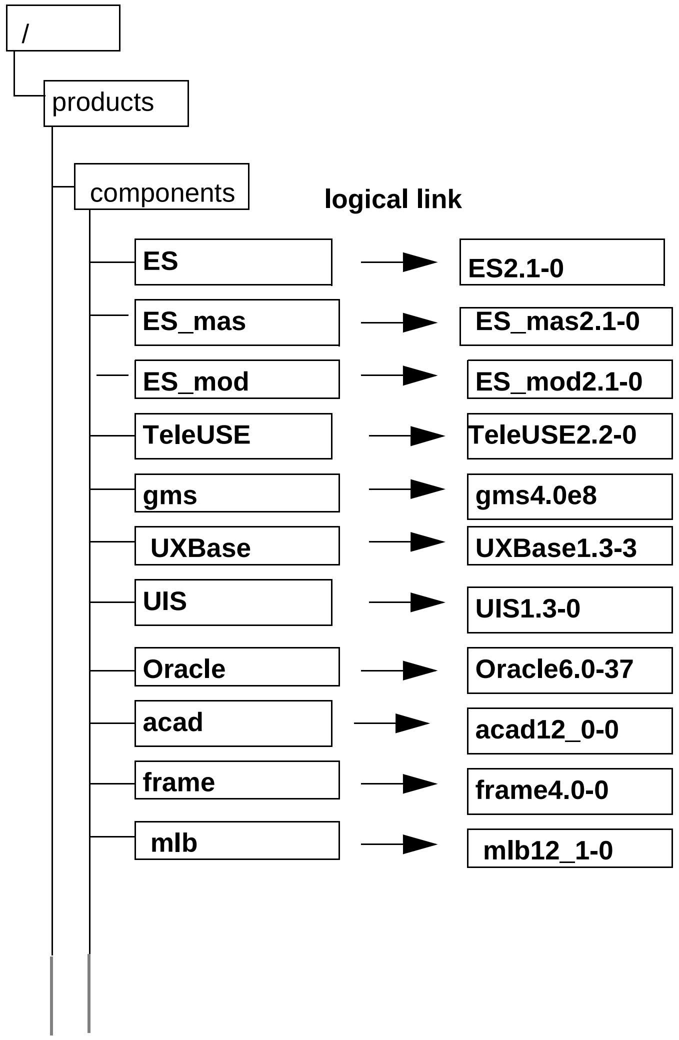


Figure A-2. Products Structure

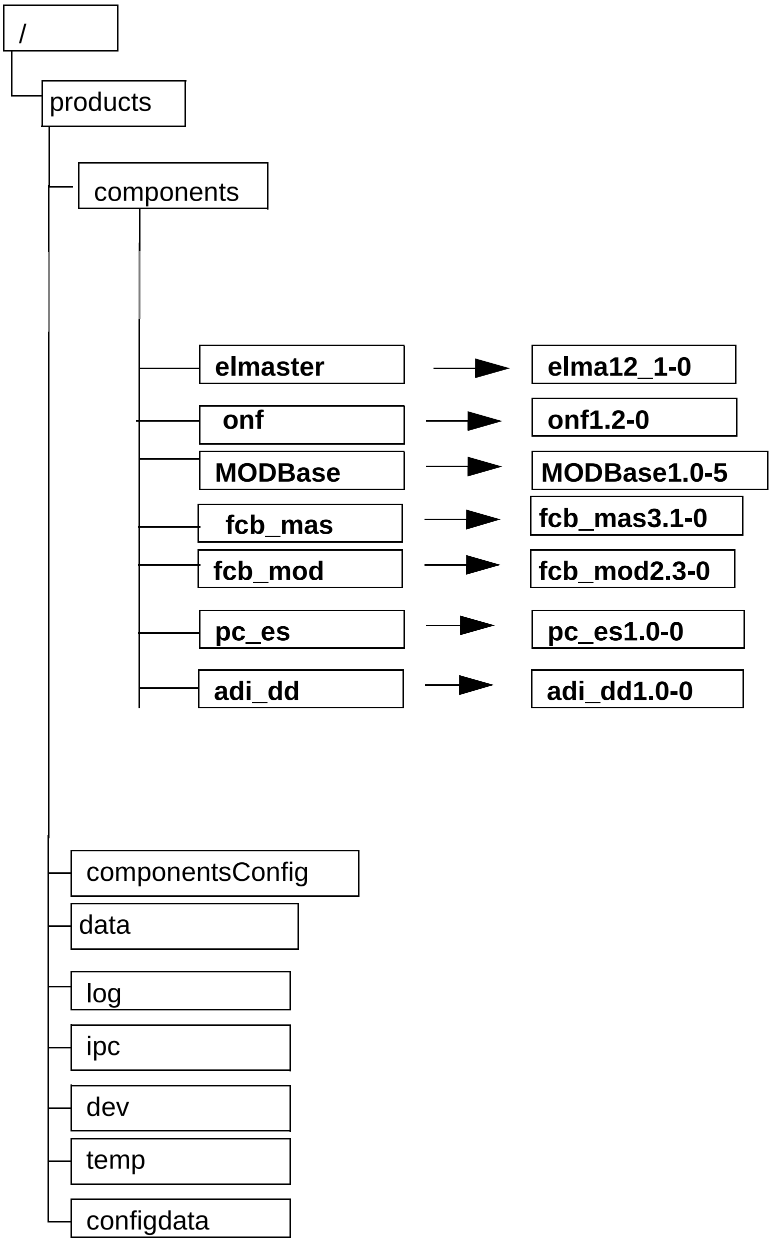


Figure A-3. Configuration Files

A.3 Data Directory Structure

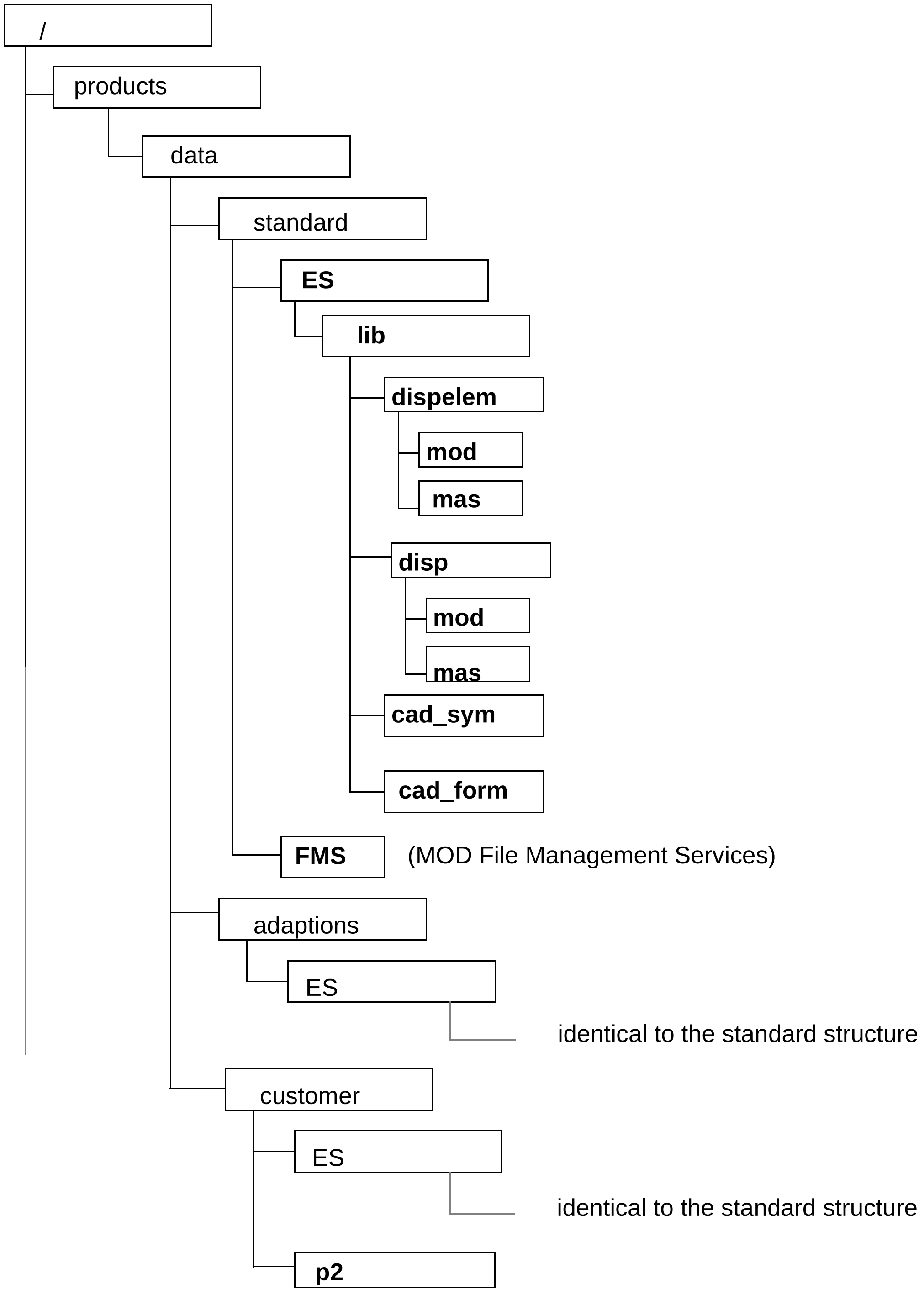

/
products
data
standard
ES
lib
dispelem
mod
mas
disp
mod
mas
cad_sym
cad_form
FMS
(MOD File Management Services)
adaptions
ES
identical to the standard structure
customer
ES
identical to the standard structure
p2

Figure A-4. Data Directory Structure

- /products/data/standard/ES

  Contains AS500ES specific data (typically libraries) provided with the system. There is only one substructure defined so far for libraries "/lib". The structure below "/lib" follows the general library concepts of AS500ES.

- /products/data/adaptions/ES

  Contains BA, BU, LIPA specific data (typically libraries). There is only one substructure defined so far for libraries "/lib"; the structure is identical to the ../standard/ES.

- /products/data/customer/ES

  Contains customer specific data (typically libraries). There is only one substructure defined so far for libraries "/lib"; the structure is identical to the ../standard/ES.

- /products/data/customer/p2.

  Communication directory for P2-client/server applications (PAGIS project).

A.4 The proj Directory Structure

The "proj" substructure contains the different projects "<project 1>" to "<project n>" where the engineering data is stored, Figure A-5. The directory structure "proj_adm" is for superior project data like Function Chart Builder libraries or, for example, ORACLE Forms which are common for all engineering projects. Also, the project data is stored here if a project was saved by the administration tool. The mod_std and mas_std directory structures are used by the MOD 300 and Master standard projects. These directories are also created at installation time. It is up to the user to delete one of them if not needed. The standard projects are used to maintain the Object Type Definitions used to create engineering projects and to maintain library data.

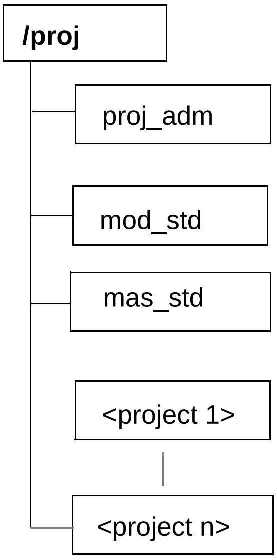


Figure A-5. proj Directory Structure

A.5 The proj_adm Directory Structure

The structure is considered to keep common project overall data, for example, libraries for Function Chart Builder, Diagram Builder, log files from the administration tool and so on. See Figure A-6.

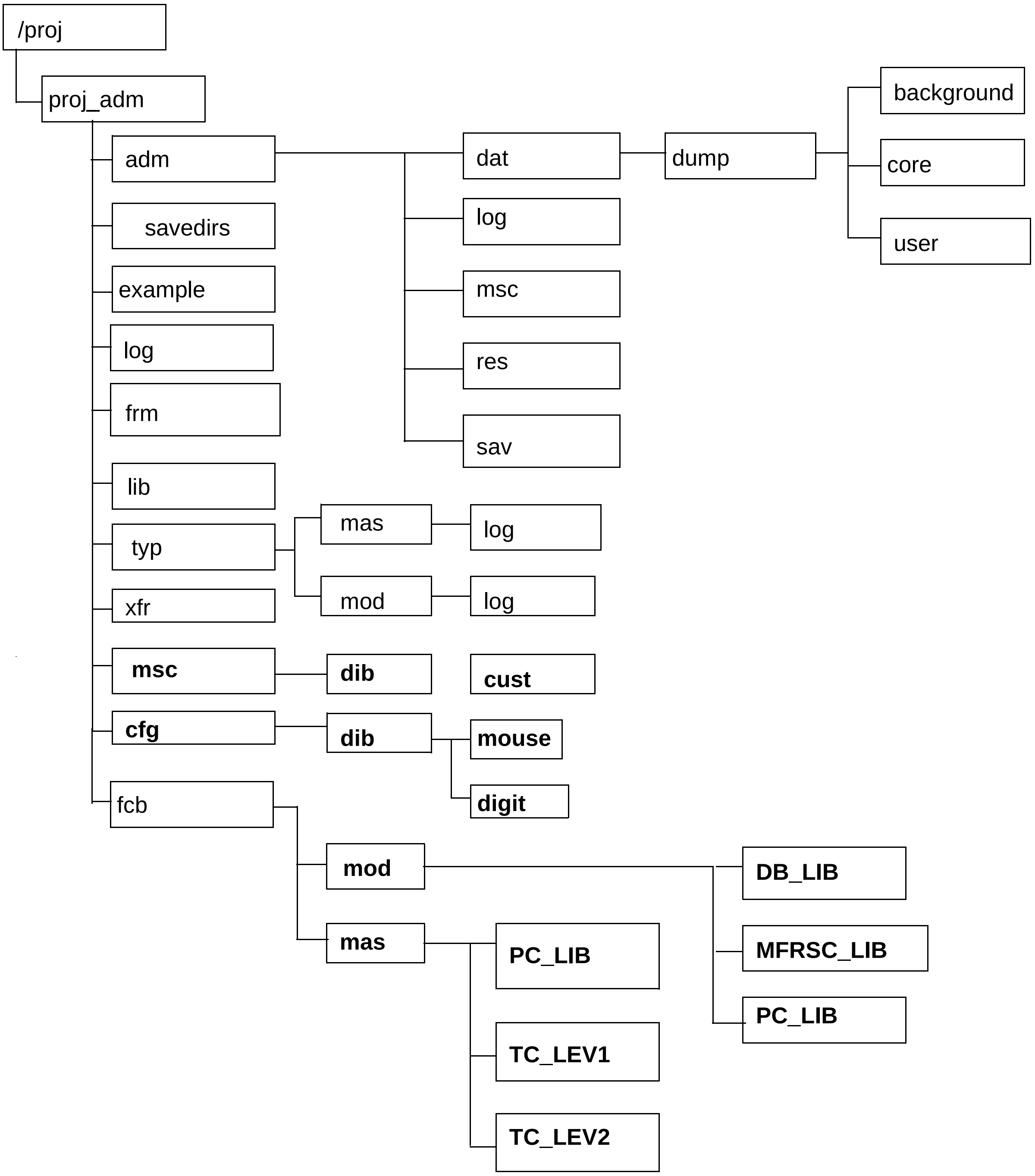


Figure A-6. proj_admin Directory Structure

A.6 The <project> Directory Structure

Project specific data for example (/eiu) data for import into the ES data base or data which was exported is stored here as shown in Figure A-7. Also stored here are libraries, data from Display Builder, the node (for example, MP200) directory structure and for every project member a directory with sub structures. See Figure A-8 through Figure A-10.

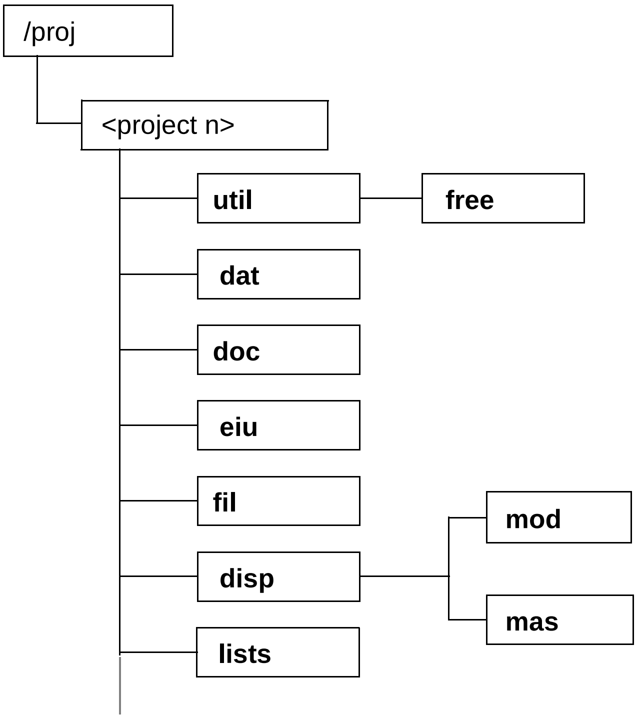


Figure A-7. Project Specific Data in <project> Directory Structure

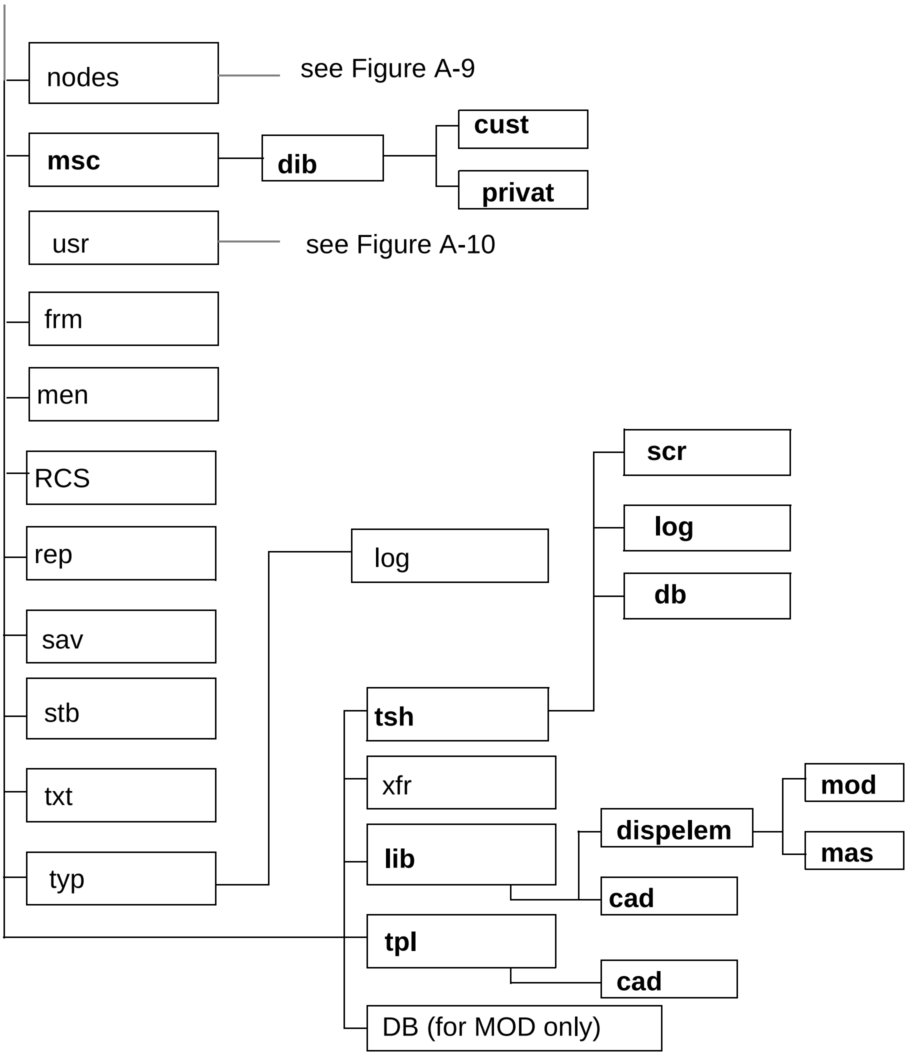


Figure A-8. Library Data in <project> Directory Structure

- /proj/<project>/nodes

  This is the entry point for node system data. The node system data is considered to be local to each project. So in another project there could be another independent node system structure. The substructure is based on DCS nodes. There is one subdirectory structure per node with the directories like PCDATA, DBDATA, USER, SRCE and so an. For example, backup can be done either in the directory "backup" see below or into the subdirectories "USER or SRCE" which is up to the user

- /proj/<project>/nodes/backup

  This is a central directory for backup of nodes in the plant network which are identified by a network, node number (for example, Advant Controllers).

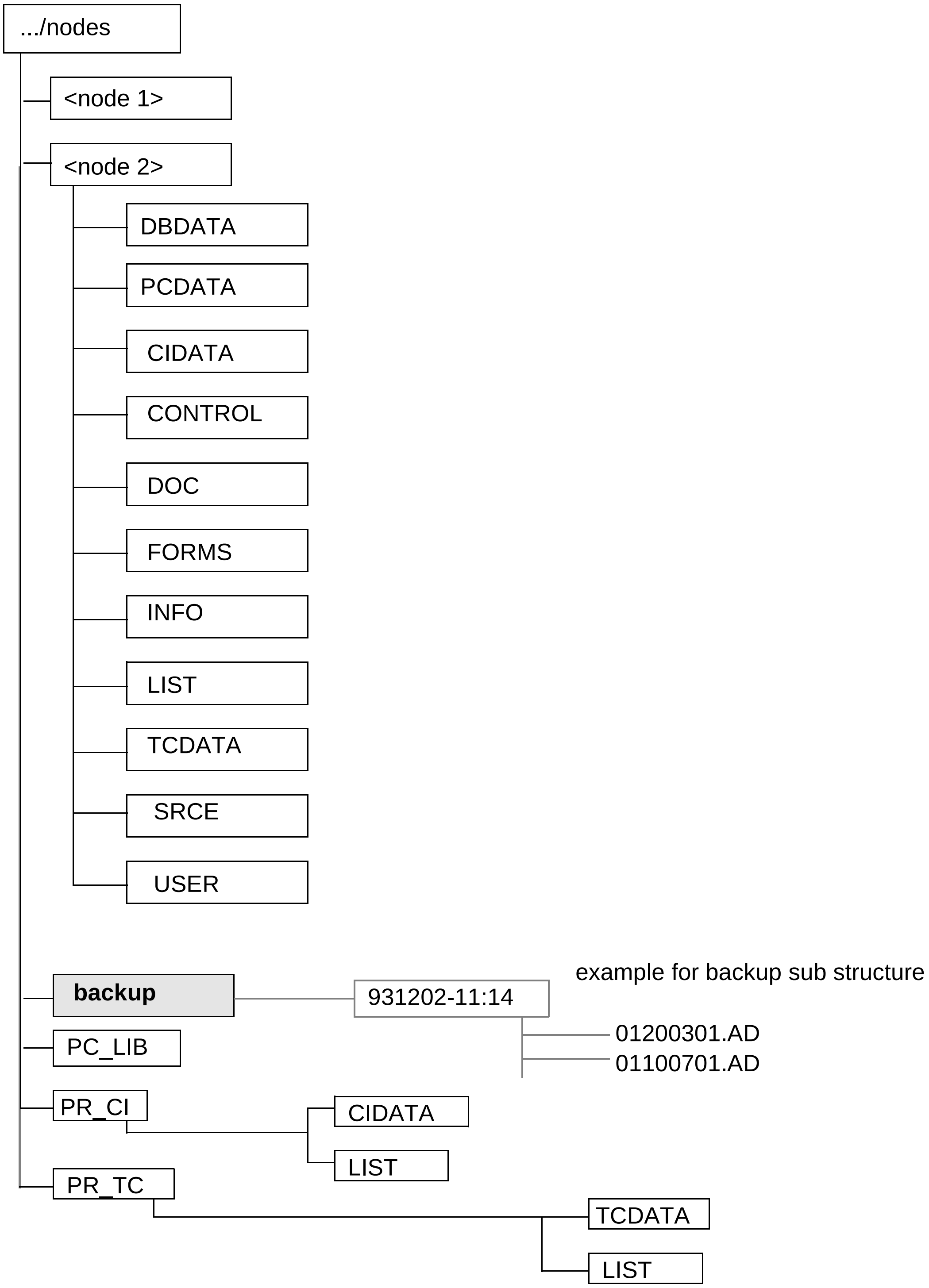


Figure A-9. Node Directory Structure

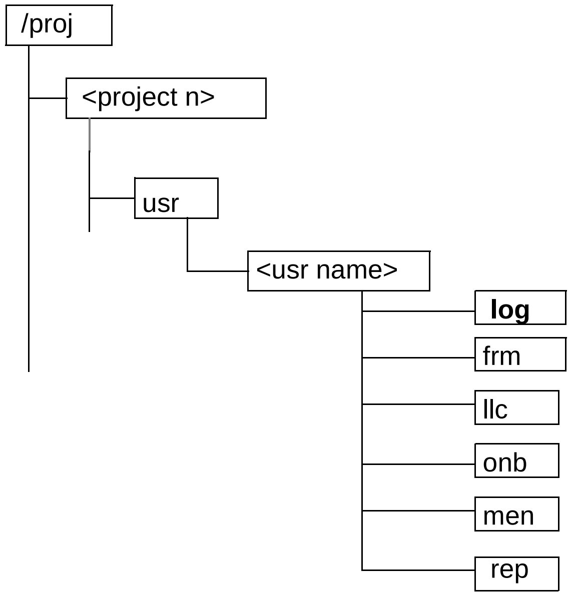


Figure A-10. Project Member Directory Structure

A.7 AS500ES Environment Variables

The environment variables are set at log-in time. This means for common variables at HP login when the AS500ES user logs into the workstation, project and user specific variables will be set when the AS500ES user logs into the engineering station via Structure Builder.

A.8 Environment Variables Referring to Products Structure

```
C_BIN=/products/components
C_BIN_CAPE=/products/components/{ES_mas,ES_mod}
C_BIN_FCC=/products/components/{fcb_mas, fcb_mod}/bin

C_BIN_PC_ES=/products/components/pc_es
C_BIN_LIBS=/products/components/ES/lib
C_BIN_ONF=/products/components/onf
C_BIN_ORA=/products/components/Oracle

C_BIN_ONF_BIN=${C_BIN_ONF}/bin

C_BIN_ONF_ETC=${C_BIN_ONF}/etc
C_BIN_ONF_LIB=${C_BIN_ONF}/lib
C_BIN_ONF_LIB_BOOT=${C_BIN_ONF_LIB}/boot
C_BIN_ONF_LIB_SYST=${C_BIN_ONF_LIB}/syst
C_BIN_ONF_LIB_CMDS=${C_BIN_ONF_LIB}/cmds
C_BIN_FRM=/products/components/frame
C_BIN_FRM_BIN=${C_BIN_FRM}/bin
C_BIN_FRM_TPL=${C_BIN_FRM}/fminit/ukenglish/Maker/Templates
C_BIN_CAPE_BIN=${C_BIN_CAPE}/bin # executables
C_BIN_CAPE_CTL=${C_BIN_CAPE}/ctl # loader ctl files
C_BIN_CAPE_FRM=${C_BIN_CAPE}/frm # forms
```

```
C_BIN_CAPE_TSH_FRM=${C_BIN_CAPE}/frm # forms TSH
C_BIN_CAPE_INC=${C_BIN_CAPE}/inc # include files
C_BIN_CAPE_LIS=${C_BIN_CAPE}/lis #
C_BIN_CAPE_LOG=${C_BIN_CAPE}/log # online log files
C_BIN_CAPE_MEN=${C_BIN_CAPE}/men # menus
C_BIN_CAPE_MISC=${C_BIN_CAPE}/msc # miscellaneous
C_BIN_CAPE_REP=${C_BIN_CAPE}/rep # reports
C_BIN_CAPE_SH=${C_BIN_CAPE}/sh # shellscripts
C_BIN_CAPE_TSH_SH=${C_BIN_CAPE}/sh # shellscripts TSH
C_BIN_CAPE_SQL=${C_BIN_CAPE}/sql # SQL scripts
C_BIN_CAPE_TMP=${C_BIN_CAPE}/tmp # temporary files
C_BIN_CAPE_TXT=${C_BIN_CAPE}/txt # text files
C_BIN_CAPE_TXT_ENG=${C_BIN_CAPE_TXT}/eng # e.g. C_ERROR text
C_BIN_CAPE_DYB_BIN=${C_BIN_CAPE_BIN}/dyb
C_BIN_CAPE_TSH_BIN=${C_BIN_CAPE_BIN}/tsh
C_BIN_CAPE_ENV=${C_BIN_CAPE}/env # Environment set up data
C_BIN_CAPE_DST=${C_BIN_CAPE}/dst # distribution dir structure
ORACLE_HOME=$C_BIN_ORA
TWO_TASK=${ORACLE_SERVER}_${ORACLE_SID}
ORAKITPATH=$C_CAPE_ALL_ADM_RES:$ORACLE_HOME/forms30
         /admin/resource
ORATERMPATH=$C_CAPE_ALL_ADM_RES:$ORACLE_HOME/forms30
         /admin/resource
FORMS30PATH=$C_CAPE_ALL_ADM_RES:$ORACLE_HOME/forms30
         /admin/resource
MENU5PATH=$C_CAPE_ALL_ADM_RES:$ORACLE_HOME/forms30/admin
         /resource
```

A.9 Environment Variables Referring to Data Directory Structure

```
C_CUS_P2=/products/data/customer/p2
C_STA_ES=/products/data/standard/ES
C_ADA_ES=/products/data/adaptions/ES
C_CUS_ES=/products/data/customer/ES

C_STA_ES_LIB=${/C_STA_ES}/lib
C_ADA_ES_LIB=${C_ADA_ES}/lib
C_CUS_ES_LIB=${C_CUS_ES}/lib

C_STA_ES_LIB_CSYM=${C_STA_ES_LIB}/cad_sym
C_ADA_ES_LIB_CSYM=${C_ADA_ES_LIB}/cad_sym
C_CUS_ES_LIB_CSYM=${C_CUS_ES_LIB}/cad_sym
C_STA_ES_LIB_CFORM=${C_STA_ES_LIB}/cad_form
C_ADA_ES_LIB_CFORM=${C_ADA_ES_LIB}/cad_form
C_CUS_ES_LIB_CFORM=${C_CUS_ES_LIB}/cad_form
C_STA_ES_LIB_DISPEL=${C_STA_ES_LIB}/dispelem/{mod,mas}
C_ADA_ES_LIB_DISPEL=${C_ADA_ES_LIB}/dispelem/{mod,mas}
C_CUS_ES_LIB_DISPEL=${C_CUS_ES_LIB}/dispelem/{mod,mas}

C_STA_ES_LIB_DISP=${C_STA_ES_LIB}/disp/{mod,mas}
C_ADA_ES_LIB_DISP=${C_ADA_ES_LIB}/disp/{mod,mas}
C_CUS_ES_LIB_DISP=${C_CST_ES_LIB}/disp/{mod,mas}
```

A.10 Environment Variables Referring to proj_adm Directory Structure

```
C_CAPE=/proj
C_CAPE_ALL=${C_CAPE}/proj_adm
C_CAPE_ALL_ADM=${C_CAPE_ALL}/adm # user ADMIN’s root dir
C_CAPE_ALL_ADM_DAT=${C_CAPE_ALL_ADM}/dat # tablespace ADMINdbf
             file
C_CAPE_ALL_ADM_DAT_DMP=${C_CAPE_ALL_ADM_DAT}/dump
C_CAPE_ALL_ADM_DAT_DMP_B=${C_CAPE_ALL_ADM_DAT_DMP}/background
C_CAPE_ALL_ADM_DAT_DMP_C=${C_CAPE_ALL_ADM_DAT_DMP}/core
C_CAPE_ALL_ADM_DAT_DMP_U=${C_CAPE_ALL_ADM_DAT_DMP}/user
C_CAPE_ALL_ADM_LOG=${C_CAPE_ALL_ADM}/log # logfiles of OMA
C_CAPE_ALL_ADM_MISC=${C_CAPE_ALL_ADM}/msc #
C_CAPE_ALL_ADM_RES=${C_CAPE_ALL_ADM}/res # for resource files
C_CAPE_ALL_ADM_SAV=${C_CAPE_ALL_ADM}/sav # export files of OMA
C_CAPE_ALL_FCC=${C_CAPE_ALL}/fcb/{mod, mas}#
C_CAPE_ALL_FCC_DB_LIB=${C_CAPE_ALL_FCC}/DB_LIB #
C_CAPE_ALL_FCC_TC1=${C_CAPE_ALL_FCC}/TC_LEV1 # BU type circuits
C_CAPE_ALL_FCC_TC2=${C_CAPE_ALL_FCC}/TC_LEV2 # BA type circuits
C_CAPE_ALL_FCC_PC=${C_CAPE_ALL_FCC}/PC_LIB # PC-Element
libraries
C_CAPE_ALL_FCC_MFRSC=${C_CAPE_ALL_FCC}/MFRSC.LIB # for MOD
C_CAPE_ALL_LOG=${C_CAPE_ALL}/log # offline log files
C_CAPE_ALL_FRM=${C_CAPE_ALL}/frm #
C_CAPE_ALL_LIB=${C_CAPE_ALL}/lib # libraries
C_CAPE_ALL_EXP=${C_CAPE_ALL}/example # examples
C_CAPE_ALL_MSC=${C_CAPE_ALL}/msc
C_CAPE_ALL_MSC_DIB=${C_CAPE_ALL_MSC}/dib
C_CAPE_ALL_MSC_DIB_CUS=${C_CAPE_ALL_MSC_DIB}/cust
C_CAPE_ALL_CFG=${C_CAPE_ALL}/cfg #
C_CAPE_ALL_CFG_DIB=${C_CAPE_ALL_CFG}/dib
C_CAPE_ALL_CFG_DIB_MOUSE=${C_CAPE_ALL_CFG_DIB}/mouse
C_CAPE_ALL_CFG_DIB_DIGIT=${C_CAPE_ALL_CFG_DIB}/digit
C_CAPE_ALL_TYP=${C_CAPE_ALL}/typ
C_CAPE_ALL_TYP_MAS=${C_CAPE_ALL_TYP}/mas
C_CAPE_ALL_TYP_MAS_LOG=${C_CAPE_ALL_TYP_MAS}/log
C_CAPE_ALL_TYP_MOD=${C_CAPE_ALL_TYP}/mod
C_CAPE_ALL_TYP_MOD_LOG=${C_CAPE_ALL_TYP_MOD}/log
C_CAPE_ALL_XFR=${C_CAPE_ALL}/xfr
```

A.11 Environment Variables Referring to proj Directory Structure

```
C_PROJ=/proj/<project name>
C_PROJ_UTIL=${C_PROJ}/util
C_PROJ_UTIL_FREE=${C_PROJ_UTIL}/free
C_PROJ_DAT=${C_PROJ}/dat
C_PROJ_DOC=${C_PROJ}/doc
C_PROJ_EIU=${C_PROJ}/eiu
C_PROJ_FIL=${C_PROJ}/fil
C_PROJ_FCC=${C_PROJ}/nodes
```

```
C_PROJ_FCC_BU=${C_PROJ_FCC}/backup
C_PROJ_FCC_PR_TC=${C_PROJ_FCC}/PR_TC
C_PROJ_FCC_PR_TC_LIST=${C_PROJ_FCC_PR_TC}/LIST
C_PROJ_FCC_PR_TC_TCDATA=${C_PROJ_FCC_PR_TC}/TCDATA
C_PROJ_FCC_PR_CI=${C_PROJ_FCC}/PR_CI
C_PROJ_FCC_PR_CI_LIST=${C_PROJ_FCC_PR_CI}/LIST
C_PROJ_FCC_PR_CI_CIDATA=${C_PROJ_FCC_PR_CI}/CIDATA
C_PROJ_FCC_LIB=${C_PROJ_FCC}/PC_LIB
C_PROJ_MSC=${C_PROJ}/msc
C_PROJ_MSC_DIB=${C_PROJ_MSC}/dib
C_PROJ_MSC_DIB_CUS=${C_PROJ_MSC_DIB}/cust
C_PROJ_MSC_DIB_PRIVAT=${C_PROJ_MSC_DIB}/privat
C_PROJ_USR=${C_PROJ}/usr
C_PROJ_FRM=${C_PROJ}/frm
C_PROJ_MEN=${C_PROJ}/men
C_PROJ_RCS=${C_PROJ}/RCS
C_PROJ_REP=${C_PROJ}/rep
C_PROJ_SAV=${C_PROJ}/sav
C_PROJ_STB=${C_PROJ}/stb
C_PROJ_TXT=${C_PROJ}/txt
C_PROJ_TYP=${C_PROJ}/typ
C_PROJ_TYP_LOG=${C_PROJ_TYP}/log
C_PROJ_XFR=${C_PROJ}/xfr
C_PROJ_LIB=${C_PROJ}/lib
C_PROJ_LIB_DISPEL=${C_PROJ_LIB}/dispelem/{mod,mas}
C_PROJ_LIB_CAD=${C_PROJ_LIB}/cad
C_PROJ_TPL=${C_PROJ}/tpl
C_PROJ_TPL_CAD=${C_PROJ_TPL}/cad
C_PROJ_TSH=${C_PROJ}/tsh
C_PROJ_TSH_SCR=${C_PROJ_TSH}/scr
C_PROJ_TSH_LOG=${C_PROJ_TSH}/log
C_PROJ_TSH_DB=${C_PROJ_TSH}/db

C_PROJ_CLASS={MOD,MASTER}
```

A.12 Environment Variables Referring to User Specific Directories

```
C_USR=${C_PROJ}/usr/<user name>
C_USR_ONB=${C_USR}/onb
C_USR_FRM=${C_USR}/frm
C_USR_LLC=${C_USR}/llc
C_USR_MEN=${C_USR}/men
C_USR_REP=${C_USR}/rep
C_USR_LOG=${C_USR}/log
```

A.13 General Environment Variables Setting

```
C_PRINT_ASCII=lp# HPLJ3Si
C_PRINT_POST=ps# HPLJ3SI
```

```
C_PRINT_HPGL=hpgl# HPLJ3SI
C_PRINT_ASCII_D=dlp# HPLJ3SI
C_PRINT_POST_D=dps# HPLJ3SI
C_PRINT_HPGL_D=dhpgl# HPLJ3SI
C_PRINT_ASCII_XL=lpxl# XL300
C_PRINT_POST_XL=psxl# XL300
C_PRINT_HPGL_XL=hpglxl# XL300
ORACLE_SID=ES_DB
LANG=american.iso88591
LANGUAGE=AMERICAN_AMERICA.WE8ISO8859P1
C_LANGUAGE=01

ORACLE_SERVER=‘hostname‘

TeleUse=/products/components/TeleUSE

UXBASE=/products/components/UXBase
UXBASE_HOME=/products/components/UXBase
BSMESSAGE=${UXBASE_HOME}/include/bsSubSystem.def

ACADHOME=/products/components/acad
ACADDRV=${ACADHOME}/drv
ACADCFG=$ACADHOME
ACADSERVER=....
ACAD=”$ACADHOME;$ACADHOME/support;$ACADHOME/fonts”
MLBHOME=/products/components/mlb
ELMAHOME=products/components/elma
UIS_HOME=/products/components/UIS
ROOTPATH_FILE=${UIS_HOME}/lib/config/OS.root
SUBSTRUCTURE_FILE=${UIS_HOME}/lib/config/OS.sub
NLSPATH=$NLSPATH:${UIS_HOME}/lib/nls/%L/%N.cat
XAPPLRESDIR=$XAPPLRESDIR:${UIS_HOME}/lib/app-defaults
UISDBHELP=${UIS_HOME}/lib/nls/C/uic
GMS_HOME=/products/components/gms
SHLIB_PATH=/products/components/TeleUSE/lib
alias hawkkill=$MODBase_HOME/bin/dxusvKill
alias hawkstart=$MODBase_HOME/bin/dxusvStart
FMHOME=/products/components/frame
FRAMEUSERSD_HOST=....
C_BIN_FRM=/products/components/frame
C_BIN_FRM_BIN=${C_BIN_FRM}/bin
MODBase_HOME=/products/components/MODBase
ADMIN_ASK=NO
CS_CONFIG=/products/data/standard/cs
CS_HOME=${MODBase_HOME}
FMSPATH=/products/data/standard/FMS
C_TSH_CFG_INST_INCPOS=ERROR
UserEditor=vuepad
C_AS100ES=/users/as100es# communication directory between AS100
and AS500 to exchange data files
```

Appendix B Glossary

B.1 Abbreviations

AC	ABB Advant Controller
AF	ABB Advant Fieldbus
AMPL	ABB MasterPiece Language
ANSI	American National Standards Institute
API	Application Programming Interface
AS	ABB Advant Station
AB	Application Block (object-type)
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAPE	Computer Aided Plant Engineering
CASE	Computer Aided Software Engineering
CAST	Computer Aided Sales Tool
CC	Control Circuit
CCITT	(french) Comité Consultatif Internationale de Télégraphie et Téléphonie
CI	CIrcuit (Master)
CM	Control Module
COTD	Common Object Type Description
CPS	Configuration tools for Process Station
CPU	Central Processing Unit
CT	Configuration Tools
DAT	Digital Audio Tape
DB	DataBase
DBMS	DataBase Management System
DCS	Distributed Control System
DDL	Data Definition Language
DE	Display Element
DEB	Display Element Builder
DED	DB-Element Description (ASCII text file) (Master)
DIB	DIagram Builder
DIN	(german) Deutsches Institut fuer Normung
DML	Data Manipulation Language
DOB	Document Builder
DOS	Disk Operating System (operating system for IBM PC and compatible computers)

DS'xx','yyyy'	Master Board 'xx' of type 'yyyy' (Master)
DTP	DeskTop Publishing
DXF	Data eXchange Format; de facto standard ASCII-format (AutoCAD®)
EDIB	Electrical DIagram Builder
DYB	Display Builder
EIU	AdvaBuild Export / Import Utility
EMC	ElectroMagnetic Compatibility
ES	Engineering Station
ET	External Tools
FAT	Factory Acceptance Test
FB	Function Block
FC	Function Chart
FCB	Function Chart Builder
FCE	Function Chart Editor (Master)
FD	Functional Description
FES	Field Engineering System (MA 220)
FUP	(german) FUnktionsPlan (english FC)
GCF	Group Control Function
GMA	(german) Gesellschaft für Meß- und Automatisierungstechnik
HLC	AdvaBuild High Level C-Interface
HP-GL	Hewlett Packard Graphic Language
HP-UX	Hewlett Packard UNIX operation system
HW	HardWare
IAT	Indirect Access Tool
I&C	Instrumentation and Control
IEC	International Electrotechnical Commission
IGES	Interactive Graphic Exchange Specification
IMS	Information Management System
I/O	Input / Output
IP	Interchange Protocol
IPA	International Process Automation
ISO	International Standardizing Organization
ITC	Instantiated Type Circuit (Master)
ITS	Instantiated Typical Solution
IU	Instance Unique
kB	KiloByte
LAN	Local Area Network
LF	Logical File (Master)
LLC	AdvaBuild Low Level C-Interface

LR	Logical Record (Master)
LSE	Language Sensitive Editor
LTD	Link Type Definition
MA	ABB MasterAid (Master)
MADB	MasterAid DataBase handling (Master)
MAPC	MasterAid PC-element handling (Master)
MATC	MasterAid Type Circuit handling (Master)
MATE	MasterAid 220 communication board (Master)
MB	ABB MasterBus (Master)
M&C	Measurement and Control
MDB	Master DataBase (Master)
MFB	ABB MasterFieldbus (Master)
MMC	Man Machine Communication
MMI	Man Machine Interface
MMS	Manufacture Message Specification
M&K	(german) Motor- und Komponentenliste
MP	ABB MasterPiece (Master)
MRM	Motif Resource Manager
MSK	(german) MeßStellenKennzeichen
MV	ABB MasterView
MVI	Multi Vendor Interface
MXB	Memory eXpansion Bus (Master)
NAMUR	(german) NormenArbeitsgemeinschaft für Meß- Und Regelungstechnik in der chemischen Industrie
NFS	Network File System
NS	Node Structure
OBJECT_ID	OBJECT IDentification
ODB	Object DataBase (internal DB of FCE)
OEM	Original Equipment Manufacturer
OID	Object Identification (within CDB)
OOP	Object Oriented Programming
OS	Operator Station
OS	Operating System
OSF	Open Software Foundation
OSF/motif	user interface defined by OSF
OSI	Open System Interconnection
OTD	Object Type Definition
PC	Personal Computer
PC	Process Control

PCPGM	Process Control ProGraM (Master)
PDB	Plant Documentation Builder
PE	Primitive Element
PI	Process Interface
P&I	Pipes and Instrumentations
PLC	Programmable Logic Controller
PLS	(german) ProzeßLeitSystem (english DCS)
PLT	(german) ProzeßLeitTechnik
PMR	Post Mortem Review
PPC	Power Plant Control
PS	Process Station
PSL	Plant Specification Language
PU	Project Unique
RDBMS	Relational DataBase Management System
RDBS	Relational DataBase System
RCOM	Remote COMmunication protocol
RMC	Rolling Mill Control
RMS	Record Management System
RS	Requirement Specification
RS 232	Specification for serial interface
RTU	Remote Terminal Unit
RW	SQL Report Writer (ORACLE)
RWM	Read Write Memory
SC	Source Code
SCF	Single Control Function
SDE	Software Development Environment
SG	Style Guide
SIG	SIGnal
SL-GMS	Sherill-Lubinsky Graphical Modelling System
SOL	SOLution
SPS	(german) SpeicherProgrammierbare Steuerung (english PLC)
SQL	Standard Query Language
STB	STructure Builder
SV	ABB SuperView (Master)
SW	SoftWare
TAG	instrumentation loop, measuring point (german PLT-Stelle)
TAS	Total Automation System
TC	Type Circuit
TCP	Transport Control Protocol

TCP/IP	Transport Control Protocol (Internet Protocol)
TEB	Template Builder
TS	Typical Solution
UI	User Interface
UIL	User Interface Language
UIMS	User Interface Management System
UIS	User Interface System
VAX/VMS	DEC operating system
VDU	Video Display Unit
VNS	(german) VerfahrensNeutrale Schnittstelle, standard CAD-interface
VUE	Visual User Environment
WS	WorkStation
WYSIWYG	"What You See Is What You Get"
X11	X-windows implementation version 11.

B.2 Terms

activities

Activities are tasks, which have to be done during *project phases*. The sequence of performed *activities* is defined by the *activity plan*, which can be adopted to different *project types*.

activity plan

The *activity plan* describes:

- what *activities* are defined for a project
- which inputs must be available for an *activity*
- which outputs are produced by an *activity*
- in which time period an *activity* can take place
- how much efforts an *activity* takes
- who is responsible for an *activity*.

ADMIN

The ADMINuser is the AdvaBuild database administrator, and has unrestricted access to the whole AdvaBuild system [15].

administration component

The *administration component* is a component of the Structure Builder. It allows the user to perform administrative project tasks, such as administration of project members, their access rights etc. [15].

application block (APPL_BLOCK)

An *application block* means a block of instructions stored in a process station to realize and to automate the addressed process functions. The application block can be described by function charts, ladder diagrams or structured text. In the Master system only PC-elements are available to built up application blocks.

aspect

An *aspect* is covered by one or more AdvaBuild object instances, e. g. the control logic data aspect is represented by one or more APPL_BLOCK objects [9].

attribute

Each object type is described by *attributes* (e. g. the terminals of a database element such as IMPL or TYPE).

block diagrams

Block diagrams are overview diagrams using block symbols [3].

builder

A tool in AdvaBuild are called *builder*.

browser

see "*tree presentation*".

call name

The *call name* is the name of a database element of a MasterPiece 200 or a Advant Controller 110 station. As a general rule the call name of a Master database element is identical to the name of the corresponding object type in the AdvaBuild database.

CAPE (Computer Aided Plant Engineering)

CAPE is a concept for plant engineering. CAPE provides a framework for various types of tools, called builders, including configuration tools.

channel

Devices making a certain function available more than one time can be seen as a combination of a corresponding number of virtual (sub-)devices making this function available only one time. Each of these virtual (sub-) devices is called a *channel*, for example, an I/O board consists usually of 16 or 32 channels.

child object

A *child object* is an object of the tree refining the preceding level's object. The refined object is called parent object.

commit

Commit is the process of storing new, changed or deleted objects from the form internal work space to the corresponding table in the database [10].

CONFIG

For each table space a user CONFIG is established, who can configure the system in detail [15].

configuration tools (CT)

Configuration tools are tools for the arrangement of software units. In contrast to engineering tools, configuration tools can have on-line access to the DCS. It is distinguished between configuration tools for process-station (CPS) and IMS/OS configuration tools.

connected mode

In the *connected mode* tools have direct access to the DCS-database in the target system(s), but simultaneously the AdvaBuild engineering database is updated.

configuration component

The *configuration component* is a component of the Structure Builder. It allows the user to adopt the AdvaBuild system in a predefined framework [15].

control logic

Control logic is the former expression for an application block, see “application block”.

Control-system[-oriented] structure (# structure, DCS[-function oriented] structure)

This structure is dependant on the DCS and represents the actual DCS configuration.
The objects of the *Control-system-oriented structure* represent the control system functions.

converter

A *converter* is a tool-specific interface program, which updates data between tool-specific databases and the common data within the AdvaBuild database in both directions.

database

- AdvaBuild database

  The *AdvaBuild database* administrates all information concerning the delivery project and ensures consistency of data. The AdvaBuild database is a distributed database, which means, that information can be located on different computers in a network. The AdvaBuild database is based on the ORACLE relational database.

- Master database:

  The *Master database* is the database of the Master process control system. The contents of this database (i. e. Master DB-elements or Master PC-elements) are distributed among the process stations of the control system.

- tool-specific database:

  A *tool-specific database* is a specific database of a special Builder and exists in addition to the AdvaBuild database. Update of data between a tool-specific database and the common AdvaBuild database is done via converter.

- Object database:

  The *object database* is the tool-specific database of the Function Chart Builder (FCB). It contains the PC-elements for AMPL programming.

database index

In the header of all database elements of the Master database there are two numbers in brackets, separated by a point (logical file.logical record). Together these form a unique identity for this database element, the so-called *database index*. The database index can be used to identify the element within a MasterPiece station[14].

DB-elements

DB-elements provide a software concept to view and modify certain data of a MasterPiece station. The contents of the database are grouped into DB-elements. Each DB-element contains the information necessary for a complete function, such as an analog input signal or printer set-up data. It is distinguished between different groups of database elements [10]:

- MP200 DB-elements:
  - signal DB-elements (e. g. AIS, AOS),
  - function unit DB-elements Group1 (e. g. AIC, AOC),
  - function unit DB-elements Group2 (e. g. GENBIN, GENCON),
  - function unit DB-elements Group3 (e. g. MANSTN, MOTCON, PIDCON).
- Advant Controller 110 DB-elements:
  - signal DB-elements (e. g. AIS610, AIS620).
- BU specific DB-elements:
  - BU Fire and Gas (e. g. FD, FI),
  - BU Pulp and Paper (AAB_FILE),
  - BU Drives (e. g. DCU, TIME_DI).

DB-objects

The data of DB-elements are stored in the relational ORACLE database of AdvaBuild as objects.

default value

When creating a new object, all attribute fields get the *default values* assigned as defined in the AdvaBuild database. The user is able to modify the default values before the data is committed to the database [10].

designation code

The *designation code* is used to designate all items, which have to be handled, and therefore the designation code must be unambiguous. Designation code depends on the designation code system. This system normally is standardized by ANSI-, IEC-, DIN- or customer standards.

Diagram Builder (DIB)

The *Diagram Builder* uses AutoCAD® as base system to provide CAD functionality within AdvaBuild. AutoCAD® is a powerful and multipurpose drawing tool, which allows the design of free drawings and diagrams. Different libraries are accessible by the design applications.

The Diagram Builder is recommended to be used for overview diagrams, P&I diagrams and hook-up diagrams.

The Diagram Builder can be enhanced by adding the Electrical Diagram Builder, which is further subdivided into:

- Electrical Diagram Builder Circuit (EDIB Circuit), used for circuit diagrams
- Electrical Diagram Builder Layout (EDIB Layout), used for layout diagrams.

Display Builder (DYB)

The *Display Builder* is a configuration tool for the definition and the generation of process displays for the operator station.

Display Element Builder (DEB)

The *Display Element Builder* is used to design the display elements (dynamic graphic objects), which can be included in a display by the Display Builder.

Document Builder (DOB)

The *Document Builder* uses FrameMaker® as basis software to integrate free styled documents into AdvaBuild documentation and structure. It provides full-featured Desktop Publishing (DTP) functions including a complete word processor, a graphic toolbox for manipulating technical graphics, importing capabilities for text and graphics and easy-to-use page and document design.

document object

Document objects are predefined object types for documents (e.g. hook-up, P&I diagram, text, drawing).

Documentation structure (& structure)

This structure defines packages of application documentation. Different documentation packages may be defined for example, for delivery, commissioning and so on.

Electrical Diagram Builder Circuit (EDIB circuit)

The Diagram Builder permits functions to create circuit diagrams. They are used to show in a function oriented way the electrical connection between electrical components.

Electrical Diagram Builder Layout (EDIB Layout)

The *Electrical Diagram Builder Layout* permits functions to create layout diagrams. They are used, for example, to show the arrangement of cubicles, control desks and other main equipment in customer rooms. Within layouts distances and sizes are defined.

engineering tools

Engineering tools are tools which are needed for plant engineering (field, switch gear, DCS). They can only work off-line.

File-objects

File-objects are objects that contain a file or a “byte stream”.

form

A *form* is a table oriented view on data. AdvaBuild uses ORACLE forms as basis tool.

field

A *field* represents a column or a data entry area of a form [10].

FORMS (ORACLE ~)

FORMS is an adaptable application package for the entry of data into the ORACLE database.

FrameMaker®. The desktop publishing product FrameMaker® is used in AdvaBuild as Document Builder.

function

AdvaBuild distinguishes between the following kinds of functions:

- I/O-function:

  The *I/O-function* is a term that describes a single I/O channel’s function. In the Master system this description is done by Master database elements e. g. AIS, DIS, ...

- device-function:

  A *device-function* is a term that describes a function of a device or a device channel (usually a device within the field of process stations).

- control-function:

  A *control-function* is a term in AdvaBuild for a part of a controller’s application program (e. g. application block, functional unit).

functional unit

Functional units are Master database elements with integrated man-machine interface for MasterView.

function chart

The *function chart* is the presentation of a control function in the form of a diagram that contains function blocks connected by lines that represent signals sent between the function block terminals.

Function Chart Builder (FCB)

The *Function Chart Builder* is designed in AdvaBuild for off-line graphical configuration of PC-elements in the case of Master applications and for documentation of application programs by use of function blocks. The Function Chart Builder may be used for different target systems asMasterPiece200/1 or Advant Controller.

Function Chart Editor (FCE)

The *Function Chart Editor* is a standard MasterAid Software product for the programming and documentation of MasterPiece process stations. The Function Chart Editor is designed for off-line configuration of PC-elements, DB-elements and type circuits. It is based on IBM PC compatible computers running MS DOS operating system [6].

hierarchy

Objects within AdvaBuild are structured in hierarchies according to the different views a user may have on a subjects matter (e. g. functional oriented view, location oriented view, control oriented view). The object *hierarchy* is described with parent-child relationships

hook-up diagram

A *hook-up diagram* shows the arrangement of components like pipes, valves, sensors and their physical mounting [3].

import / export functionality

Import / export functionality allows to take over engineering data from outside AdvaBuild by use of ASCII files or extract AdvaBuild data into an ASCII file for external use.

instance button

The *instance button* (I-button) is an indicator, which shows that an object in the =, #, + or & structure belongs to an instance of a Typical Solution.

instantiated Typical Solution (ITS)

An *instantiated Typical Solution* is the result of instantiating a Typical Solution, i. e. by filling all parameters covered by a Typical Solution with problem specific values and explicit instantiation process objects are created in the AdvaBuild structures according to the Typical Solution.

instantiation

Instantiating is the act of creating an object. Different types of instantiations are to distinguish:

- instantiation of an object type:

  An instance is one of many objects created from the same object type.

- instantiation of a type Typical Solution:

  The Typical Solution is instantiated in two steps. First in the library structure an object of the object type Typical Solution (e. g. a temperature control (instantiation of an object type) is created and in a second step the Typical Solution object is instantiated (instantiated Typical Solution).
- instantiation of the object type L_DEV:

  The object type L_DEV is instantiated in two steps. First in the library structure an element of the object type L_DEV (a physical device) is created. It contains all common data of the devices of this kind. In a second step the adaptation to an application problem is done via references to the corresponding object L_DEV in the library structure.

internal key

The *internal key* is an unique, internal identifier of an AdvaBuild object. The internal key cannot be modified.

item designation

The *item designation* is the designation of the DB-elements and PC-elements in a MasterPiece station. The item designation is created automatically during configuration.

library

Within libraries predefined standards are administrated, e. g. Typical Solutions, device data, text, graphical symbols etc. Library objects are referenced by the instances.

Library structure

This structure defines a structure for all objects representing standards. The *Library structure* contains e.g. Typical Solutions objects.

link

Links are the database implementations of the relations between objects (e. g. a signal in the functional structure is connected to a channel of a board element object in the #-structure).

link handling

Link handling is a component of the Structure Builder that allows the user to create links between objects in the AdvaBuild system [15].

link type

Depending on the relations different *link types* have to be distinguished:

- link type HAS_SIGNAL_ELEMENT is used to assign the channels of process I/O board element object to signal element objects (Links of this type can not be deleted, but they can be opened again)
- link type HAS_CONNECTION_TO is used to assign the connections between signal element objects to process control programs, it is created automatically
- link type IS_DOCUMENTATION_OF is used to connect a documentation package to

objects

- link type IS_IMPLEMENTATED_BY is used to determine on which device a function is implemented
- link type HAS_LOCATION is used to assign electrical devices or sublocations to mounting locations. A HAS_LOCATION link may for example represent a slot or position within a mounting rack
- link type references Typical Solution is used by Typical Solutions

list object

A *list object* is an object in the list presentation.

list presentation

The *list presentation* is a table oriented view on objects as alternative to the tree representation.

Location[-oriented] structure (++, + structure)

This structure defines the location of an item in the plant, so that it can be easily found in the real plant. This structure consists actually of two different structures:

- The "topographical" structure is the superior structure (++ structure). This structure describes the physical location of the plant in different levels, like plant areas, buildings, floor, room, coordinates.
- The independent substructure to the topographical structure is the "mounting location" structure (+ structure). This structure describes the so called constructional units (cabinets, desks, racks). The low level in this structure is represented by the devices itself [1].

logical file

The *logical file* (the first part of the database index) is the file number in the Master database, where a database element is stored [14].

logical record

The *logical record* (the second part of the database index) is the record number in the file (logical file), where a database element is stored [14].

loop

A *loop* is the composition of different instrumentation devices, I/O channels, control functions, field devices and man machine interfaces over several signal paths e. g. measuring path and actuating path.

Low Level C interface (LLC)

All access to the AdvaBuild database are done by use of routines and procedures in the Low Level C interface. The *Low Level C interface* is an application specific layer on the top of the AdvaBuild database.

multiple element mode (multi element form, multi object form)

In the *multiple element mode* of the Template Builder multiple objects of the selected (opened) type are presented.

network

A *network* is a communication bus for internode data transfers. There are two types [8]:

- control network

  A *control network* is a proprietary communication network. The system limits control network traffic to maintain deterministic message transfer times.

- plant network

  A *plant network* provides an open architecture for transferring all types of data. No limitations on plant network traffic exist.

node

Any logically addressable unit, which is directly connected to the plant or control network is called a *node*. Examples are process stations, operator stations [8].

node structure (NS)

The *node structure* describes how the circuits mutually relations are to be defined for the whole or a part of a MasterPiece 200 system. A node structure can be seen as an overview documentation of the process control program and its database [8].

object

Within AdvaBuild all items are stored as objects in the AdvaBuild database. An *object* may be a real plant item or an descriptive item. There are system application objects, DCS application objects (objects representing control functions e. g. DB-elements) and system objects (objects representing hardware components e. g. nodes, boards and communication links). An object can represent a complete plant, a cabinet, a signal, a cable etc. All objects have parent-child relations. Objects can be represented in tree structure or in list structure. An object can have attributes and links to other objects. On objects can be executed different activities such as insert, move, copy or delete. An object is the instantiation of the corresponding object type.

object identification

The *object identification* is the user's only way to identify a AdvaBuild object. The object identification is user definable, but it has to be unique.

object type

All objects of the same kind (e. g. of the same characteristics such as attributes or rules) are described under a common generic term, the *object type*. Each object within the AdvaBuild structures represent a concrete instance of an object type and is identified by an unique object identification. By the object type it is defined, what tools can act on an object instance, how the object can be used within the tree structures and what link types can be defined. An object type is defined by an object type name, a list of attributes and a list of actions on that object. For the different object types see [2].

off-line

Off-line means not physically connected to the run-time or target system or connected but not working directly against the functions and data of these systems [1].

on-line

On-line means physically connected and working directly against the functions and data of the target or run-time system. On-line tools have direct access to the database(s) in distributed control system(s) and to the AdvaBuild database. Configuration, service and maintenance tools are of that type [1].

ORACLE

ORACLE is the platform for the common project database within AdvaBuild. It is a portable, distributed and open database management system that delivers high performance, continuous operation and support for very large databases. ORACLE leads the industry in both performance and capability.

parameter

Parameter are used at all places within a definition of a Typical Solution, where problem specific values shall be filled in when the Typical Solution is instantiated.

parameter list

The *parameter list* is the interface of a Typical Solution. It contains the parameters, which are filled, before the Typical Solution is instantiated [9].

parent object

A *parent object* is an object of the tree structure followed by child objects, placed on the succeeding level. Corresponding parent / child objects are automatically connected.

part

Several attributes of an object type may form a logical unit, a so-called *part*.

PC-elements

PC-elements are the smallest units of the PC programming language AMPL. Each element performs a complete function, such as a logical AND gate, a timer or a feedback control element. Each PC-element has its own call name, that enables it to be called regardless of the computer system the element is in. Many of the elements have call parameters, which allow size, function, data type and other properties to be determined [4]. In the scope of AdvaBuild the PC-elements are stored in the object database of the Function Chart Builder.

PC-program

A MasterPiece application program written in AMPL is referred to as a *PC-program*.

Plant-function[-oriented] structure or Function[-oriented] structure (= structure)

This structure describes the function realized with different process equipment and is mainly independent of the DCS [1]. Higher level objects describe functions like a plant or subplant, lower level objects describe functions like an analog input signal.

Plant Documentation Builder (PDB)

The *Plant Documentation Builder* supports the definition and administration of document packages in plant documentation structure and generates the required output. The definitions of documentation packages are dependent on subjects and applications.

project phases

A project is divided into different *phases* according to the IPA project model [1]:

1. Tendering
2. Start-up
3. Basic engineering
4. Detailed engineering
5. Assembly & Integration
6. Testing
7. Delivery
8. Installation
9. Commissioning
10. Close down
11. Maintenance.

project types

Within IPA different scopes of supply for project, concerning hardware delivery (boards, stations, actuators, sensors), software delivery (base SW, application SW), and responsibility (function, throughput, product quality) exist. Out of that possible combinations typical project types may be defined for IPA projects:

type	characteristic	scope of supply
A	OEM business	components
B	system business	cubicles incl. base SW
C	control function business	application SW based on detailed specifications
D	automation business	incl. actuators, sensors and drives
E	process function business	incl. responsibility concerning production and product quality
F	turn-key-project	incl. building and infrastructure, mechanical design

AdvaBuild supports all types of process automation projects besides type F. MasterAid can cover project types A to C, maybe partly with reduced functionality or comfort.

Real Time Accelerator Board (RTA)

The *RTA board* acts as an interface between Advant Station 500 series Engineering Station (the HP UX workstation) and DCS control network, as a means to communicate with a MasterPiece 200 station [11]. The RTA-board provides on-line working to the MasterPiece 200 database

READER

The user *READER* has only read access to the AdvaBuild data [15].

section

A *section* is a application specific part of a type circuit.

signal designation

The *signal designation* identifies a signal [12].

signal reference designation

The *signal reference designation* carries the information where the signal comes from or where the signal has its destinations [12].

single element mode (single element form, single object form)

In this mode of operation for template displays the attributes for just the opened object are presented.

solution (SOL)

A *solution* contains instance specific data of a Typical Solution before the instantiation.

standard solution

The term "*standard solution*" in general shall be used to describe all kinds of solutions, which are used as prototype, i. e. solutions, which can be reused and adopt to an actual problem [9].

Structure Builder (STB, former Object Manager)

The *Structure Builder* is the core-part of the AdvaBuild frame system and provides the main services to users to handle the data stored within the AdvaBuild database and the tools integrated. It is the main tool to access the different tools to work on a selected object.

The Structure Builder is logically subdivided into four main components:

- Tree editor
- Link handling
- Configuration
- Administration.

structure object

The *structure objects* (e. g. FSTRUC, TSTRUC) are objects for the arrangement of a structure. They don't carry data.

structures

Structures organize AdvaBuild objects in a hierarchical tree according to:

- the function oriented view (= structure),
- the location oriented view (++ structure (e. g. plant, location, building, story) and
- + structure (e. g. room, cubicle, board)),
- the control system oriented view (# structure),
- the plant documentation (& structure),
- the library.

Prefixes according DIN, IEC, ISO can be used for the designation of the different structures.

tag

A *tag* is a composition of instrumentation and/or functions for measuring tasks and/or actuating tasks (see also "loop").

target system

The *target system* is the process control system (Master).

Template Builder (TEB, former OE, PSS)

The *Template Builder* is a generic tool for the input of objects and object-attributes into the AdvaBuild database. It supports the step-by-step refinement of specifications and of efficient copy- and query-functions and makes the efficient input of bulk data possible. The Template Builder is a database editor for AdvaBuild objects.

terminal

A *terminal* is a connection point on a hardware or software element that appears on a function chart diagram or a circuit diagram. A terminal is designated according to the input variable or the output variable for the hardware or software element and/or according to the designation of the connection point on the hardware element.

tools

In the scope of AdvaBuild, *tools* are called builders. There is a distinction between direct access tools, indirect access tools and external tools.

- *Direct access tools* have direct access to the AdvaBuild engineering database and are fully integrated in the user interface. AdvaBuild guarantees data consistency within the AdvaBuild database.
- *Indirect access tools* have their own tool specific database. AdvaBuild database will be updated in both directions by converters.
- *External tools* run on different hardware and have their own tool specific databases. AdvaBuild database will be accessed by the import / export function via converters [1].

tree object

A *tree object* is an object in the tree presentation.

tree presentation

The *tree presentation* of objects represents a plant structure like the function oriented structure, the location oriented structure, the control system oriented structure or the documentation structure. Objects in the tree presentation are displayed in the way of a hierarchical tree in a separate window, the tree structure window.

type circuit (TC)

A *type circuit* is an application specific standard solution for a process control problem in case of MasterPiece configuration. Type circuits are usually well tested and documented. The Function Chart Editor and MasterAid 720 support the creation and instantiation of type circuits. Type circuits can modify other objects, the instantiation of a type circuit is not protected [8].

Typical Solution (TS)

Typical solutions are previously developed and well tested solutions in the scope of AdvaBuild data. They can be reused to a clients standard problem. They are composed of a number of AdvaBuild objects. A Typical Solution may be a power supply, a temperature control or a display element. A Typical Solution can be adopted to the characteristics of an actual problem by filling it's parameters with problem specific values. The adaptation to an actual problem is done via a fixed interface. Typical Solutions are similar to type circuits but there are some differences:

- Typical solutions can cover different structures of AdvaBuild, not only the PC-program as in the case of type circuits.
- In contrary to type circuits a Typical Solution instance is strictly unchangeable. The instantiation of Typical Solutions is done by free variables such as the object identification, the attributes of a used AI-board etc. or by closing open links. Changes of the form of a Typical Solution instance or additional children following a Typical Solution instance are not allowed.
- The objects of a Typical Solution instance can appear in different structures of AdvaBuild.

typical solution interface

A Typical Solution can only be influenced via its interface. All "outside" dependencies of a Typical Solution have to be defined in its interface [9].

unconnected mode

In the *unconnected mode* tools have only access to the common AdvaBuild engineering database but not to the database of the target system.

user group

Each user of AdvaBuild has to be associated with a *user group* to define the rights he has in regard to the AdvaBuild system's table access and in the handling of certain menu items.
User groups are:

- ADMIN
- CONFIG
- READER
- WRITER.

value

Each attribute of an object type may have constraints concerning the allowed *values* distinguished by a value type [10].

variant controlling parameter

see "variant handling".

variant handling

Variant parts within a definition of a Typical Solution shall be used as mean to vary the structure of its instances. To achieve that, a set of variant controlling parameters shall control if or how often variant parts within a definition of a Typical Solution have to be included into one of the instances of the Typical Solution [9].

WRITER

The user *WRITER* has write access to the instance tables of the project, but can not create or modify object types and link types [15].

Appendix C Guideline to combine EIU with HP/UX Shellscript Programming

C.1 Introduction

AdvaBuild provides the possibility to improve the functionality for filling up the database with objects and links by specifying own "command macros". This can be used for:

- Inserting objects
- Updating existing objects
- Inserting Links
- Closing Links.

This functionality can be realized easily by using the

- AdvaBuild Export/Import Utility EIU
- HP/UX shellscript programming
- HP/UX utilities (sort, awk,...).

This provides powerful tools to the advanced user to implement in a programming like manner very efficient functionality towards the AdvaBuild database on his own responsibility.

In this guideline are described some examples for:

- Inserting/updating single objects
- Inserting/updating multiple objects
- Inserting a complex object structure using own defined "typicals"
- Inserting a flexible number of board elements
- Comfortable assignment of DB elements to I/O channels.

But the user is free to invent new applications.

C.2 Background and Setup

Command macros are made by normal UNIX shellscripts. Within the shellscripts the EIU can be called in its command line mode (see AS500ES User Manual 3BDS001674R1 Export/Import Interface).

The EIU needs normally some files:

- Selectfile (.sel)
- Controlfile (.ctl)
- Data file (.dat).

In the following examples those files are created dynamically within the shellscript files by use of the `echo` command.

Additionally the EIU needs some environment variables. The most are automatically available, if the filemanager is started via STB menu item **Utility File Manager** and is used for shellscript execution or if the shellscripts are used via the FREE object.

Following environment variables used in the shell script examples or needed by the EIU are not automatically available or preset in the correct way:

- **UID** for AdvaBuild user name
- **PWD** for AdvaBuild password

Additionally is assumed, that the basic shellscript files are located on the EIU directory, which is referenced by C_PROJ_EIU. To be able to execute the shellscripts from any other directory, this must be included in the PATH environment variable.

All these definitions most be set explicitly by the user in his shell scripts, when the FileManager is used or must be preset in the basic configuration of the FREE object, if this is used.

In the following examples relevant for use of the FileManager the definitions are done in a file *setup_env* which looks like the following example:

file *setup_env*

```
UID=mas_exp_config
export UID
PWD=mas_exp_config
export PWD
PATH=${PATH}:${C_PROJ_EIU}
export PATH
```

The definitions can be included in any shellscript by the statement:

```
. setup_env
```

and are available now for use.

Important hint:
The command in the shellscript examples calling the EIU (line starts with 'eiu') must be written in one line! This may not become clear in the source code examples.

C.3 Example of inserting a single object

This example describes a shellscript file ***obj_ins***, which can be used for inserting an object with a flexible number of attributes into the database.

All necessary files for the EIU are dynamically created by the shellscript file.

For the user all needed/produced files are available on the user directory. Those files, which are needed/produced on the EIU directory are copied automatically by the shellscript.

The shellscript file is called in the way

```
obj_ins <parent_id> <obj_id> <obj_type> <T or L.> <comment>....
....[<attribute_name> <attribute_value>]
```

Example:

```
obj_ins =A10 =A10T100 TAG T 'This is comment' TAGFUNCTION TICA++
```

Any number of attributes can be appended.

The code of the shellscript ***obj_ins*** is

```
# Creating temporary.ctl file
 echo LOAD DATA                                     > obj_ins.ctl
 echo INTO TABLE obj_temp                          >> obj_ins.ctl
 echo FIELDS                                      >> obj_ins.ctl
 echo ’(’objid POSITION’(’1:32’)’ CHAR’(’32’)’’)’ >> obj_ins.ctl
# Creating temporary .sel file
 oid=${2}
 echo CATEGORY OBJECTS                  > obj_ins.sel
 echo TASK UPDATE                      >> obj_ins.sel
 echo PARENT “’”${1}”’”                >> obj_ins.sel
 echo SELECT                           >> obj_ins.sel
 echo ATTRIBUTE obj_id “’”${oid}”’”    >> obj_ins.sel
 echo ATTRIBUTE obj_type “’”${3}”’”    >> obj_ins.sel
 echo ATTRIBUTE c_disp_char “’”${4}”’” >> obj_ins.sel
 echo ATTRIBUTE sym_name “’”${5}”’”    >> obj_ins.sel
 while [ “$6” != ’’ -a “$7” != ’’ ]
 do
   echo ATTRIBUTE ${6} “’”${7}”’”      >> obj_ins.sel
   shift
   shift
 done
 echo FROM OBJ_TEMP                    >> obj_ins.sel
 echo WHERE objid=”’”${oid}”’”         >> obj_ins.sel
#
# Inserting obj_id into .dat file
 echo ${oid}                            > obj_ins.dat
#
# Performing Import of object
#
 cp obj_ins.dat ${C_PROJ_EIU}/${UID}obj_ins.dat
 cp obj_ins.sel ${C_PROJ_EIU}/${UID}obj_ins.sel
 cp obj_ins.ctl ${C_PROJ_EIU}/${UID}obj_ins.ctl
  eiu import uid=${UID} pwd=${PWD} sel=${UID}obj_ins.sel
ctl=${UID}obj_ins.ctl dat=${UID}obj_ins.dat
 more eiu.log
 rm ${C_PROJ_EIU}/${UID}obj_ins.dat
 rm ${C_PROJ_EIU}/${UID}obj_ins.sel
 rm ${C_PROJ_EIU}/${UID}obj_ins.ctl
 cp ${C_PROJ_EIU}/${UID}obj_ins.dat.log obj_ins.dat.log
 rm ${C_PROJ_EIU}/${UID}obj_ins.dat.log
 echo Object ${oid} insert finished
#
```

C.4 Example of inserting a number of objects

The previous example can be improved in the way, that a number of objects can be inserted in one step. This means the EIU is called only once for inserting several objects. This is much faster than calling the EIU separately for each insert or update of one object.

The shellscript works in three phases:

1. Initializing the .dat, .sel and .ctl files
2. appending objects to the files (can be done in a repeating way)
3. executing the import of all appended object definitions

All needed files are created dynamical by the shellscript.

For the user all needed/produced files are available on the user directory. Those files, which are needed/produced on the EIU directory are copied automatically by the shellscript.

C.4.1 Initializing the files

Initializing the files means:

1. Creating the control file called ***obj.ctl***
2. Creating an empty select file called ***obj.sel***
3. Creating an empty data file called ***obj.dat***

The program call is ***`obj ini`***.

C.4.2 Inserting objects into files

Inserting objects into files means:

1. Appending the object definitions to the select file *obj.sel*
2. Appending the object id to the data file *obj.dat*

The program call is

```
obj app <parent_id> <obj_id> <obj_type> <T/L> <comment> [<att_name> <att_value>]
```

Remark: Any number of attribute definitions can be added defined by '<att_name> <att_value>'

Example:

```
obj app =A10 =A10T100 TAG T 'This is comment' TAGFUNCTION TICA++ STATUS P
```

C.4.3 Inserting objects into database

Inserting objects into database means to call the EIU an performing the inserts/updates related to the definitions in the .ctl, .sel and .dat files.

The inserting is done in three steps:

1. Copying all related files to the EIU directory. To get not in conflict with other users which may do similar work in parallel, the user name defined in $UID is added as prefix to each file
2. Performing the import
3. Copying the .log file back to the user directory and deleting all related files on the EIU directory

To do this step, the program is called simply by:

```
obj exe
```

A complete example in a shellscript file may look like:

```
. setup_env
obj ini
obj app =A10 =A10T100 TAG T ’This is comment 1’ TAGFUNCTION TICA++ STATUS X
obj app =A10 =A10T200 TAG T ’This is comment 2’ TAGFUNCTION TIC++ STATUS Y
obj app =A10 =A10T300 TAG T ’This is comment 3’ TAGFUNCTION TICA+ STATUS Z
obj exe
```

C.4.4 Code of shellscript obj

```
if test “$1” != ’ini’ -a “$1” != ’app’ -a “$1” != ’exe’
then
  echo  ’Short description of program use ’
  echo  ’___________________________________________________________________’
  echo  ’The program needs the environment variables’
  echo  ’  UID set to user id’
  echo  ’  PWD set to password’
  echo  ’It should be called therefore from an overordinated shellscript ’
  echo  ’which includes or defines this variables’
  echo  ’The program works in three phases’
  echo  ’Initialization:   obj ini’
  echo  ’Appending data:   obj app <parent_id> <obj_id> <obj_type> <T/L> ....’
  echo  ’                      ........<comment> [<att_name> <att_val>]’
  echo  ’Execute import:   obj exe’
  echo
  echo  ’The program can be performed from any directory’
  echo  ’(Remark: Include EIU directory in PATH variable!)’
  echo  ’It takes the .sel,.ctl.dat file from the starting directory ’
  echo  ’and copies them temporarily to the EIU directory’
  echo
  echo  ’Example:’
  echo  ’    ’obj ini
  echo  ’    ’obj app = =A10 FSTRUCT T “’”This is comment 1”’”
  echo  ’    ’obj app =A10 =A10T1 TAG T “’”This is comment 2”’” STATUS X
  echo  ’    ’obj app =A10T1 ’=A10T1&CD CI_DIA T’ “’”comment 2”’” STATUS Y LANGUAGE en
  echo  ’    ’obj exe
fi
#Initializing the object import
if test “$1” = ’ini’
then
  # Creating temporary .ctl file
  echo LOAD DATA                                     > obj.ctl
  echo INTO TABLE obj_temp                          >> obj.ctl
  echo  FIELDS                                      >> obj.ctl
  echo  ’(’objid POSITION’(’1:32’)’ CHAR’(’32’)’’)’ >> obj.ctl
  #
  # Initializing .dat file
  rm obj.dat
  touch obj.dat
  #
  # Initializing  .sel file
  rm obj.sel
  touch obj.sel
  #
  echo Object insert initialized
fi
#Importing the defined objects
if test “$1” = ’exe’
then
 cp obj.dat ${C_PROJ_EIU}/${UID}obj.dat
 cp obj.sel ${C_PROJ_EIU}/${UID}obj.sel
 cp obj.ctl ${C_PROJ_EIU}/${UID}obj.ctl
 $C_BIN_CAPE_BIN/eiu import uid=${UID} pwd=${PWD} sel=${UID}obj.sel  ctl=${UID}obj.ctl
dat=${UID}obj.dat
 more eiu.log
 rm ${C_PROJ_EIU}/${UID}obj.dat
 rm ${C_PROJ_EIU}/${UID}obj.sel
 rm ${C_PROJ_EIU}/${UID}obj.ctl
 cp ${C_PROJ_EIU}/${UID}obj.dat.log  obj.dat.log
 rm ${C_PROJ_EIU}/${UID}obj.dat.log
fi
# filling the .sel and .dat file for import
if test “$1” = ’app’
then
```

```
  shift
  oid=${2}
  echo ’’                                >> obj.sel
  echo CATEGORY OBJECTS                  >> obj.sel
  echo TASK UPDATE                       >> obj.sel
  echo PARENT “’”${1}”’”                 >> obj.sel
  echo SELECT                            >> obj.sel
  echo ATTRIBUTE obj_id “’”${oid}”’”     >> obj.sel
  echo ATTRIBUTE obj_type “’”${3}”’”     >> obj.sel
  echo ATTRIBUTE c_disp_char “’”${4}”’”  >> obj.sel
  echo ATTRIBUTE sym_name “’”${5}”’”     >> obj.sel
    while [ “$6” != ’’ -a “$7” != ’’ ]
    do
      echo ATTRIBUTE ${6} “’”${7}”’”     >> obj.sel
      shift
      shift
    done
  echo FROM OBJ_TEMP                     >> obj.sel
  echo WHERE objid=”’”${oid}”’”          >> obj.sel

  # Appending obj_id to dat file
  echo ${oid}                            >> obj.dat
  echo “Object ${oid} inserted in .sel file”
fi
```

C.5 Using nested shellscript files

Shellscript files can be nested about any level. They can be used to define for example:

- ‘typicals’ for standardized substructures
- own algorithms for multiple insert

C.5.1 Example for a TAG typical

A typical for a standardized TAG may look like the following example.

Example:

The shellscript t100 includes:

```
obj ini
obj app  =${1} ${1}_CV AIS T “Control value ${1}” UNIT mm RANGEMAX 100
obj app  =${1} ${1}_OP DIS T “Open value ${1}” NORM_POS 1
obj app  =${1} ${1}_CL DIS T “Closed value ${1}” NORM_POS 0
obj app  =${1} =${1}-U1 DEV_FUN T “Signal Transmitter 1 ${1}”
obj app  =${1} =${1}-U2 DEV_FUN T “Signal Transmitter 2 ${1}”
obj app  =${1} “=${1}&TE” TEXT L “Specification for ${1}”
obj exe
```

Remark: in this example =${1}&TE is encapsulated by ““, because & is interpreted otherwise as control character by the shell.

So the command t100 A10T1024 creates automatically as children of object =A10T1024 the objects

- A10T1025_CV with objecttype AIS
- A10T1025_OP with objecttype DIS
- A10T1025_CL with objecttype DIS
- =A10T1025-U1 with objecttype DEV_FUN
- =A10T1025-U1 with objecttype DEV_FUN
- =A10T1025&TE with objecttype TEXT.

C.5.2 Example for multiple board element creation

An algorithm for multiple board element creation may look like:

```
if test “$1” = ’’
then
echo ’  The shellscript can be used to insert a number of boards’
echo ’  in one step. ’
echo
echo ’  Call of  shellscript is:’
echo
echo ’  board_ins <node_id>  <object_type> <board_type> <start_no> <end_no>’
echo
echo ’  Remark: its assumed, that <node_id> is written without leading #.’
echo ’          This is added by the shellscript!’
echo
echo ’  Example:’
echo ’  board_ins 1U10LA10 AI DSAI_130 1 8’
echo
#
else
  obj ini
  ino=${4}
  maxno=${5}
  while [ $ino -le $maxno ]
  do
   obj app ’#’${1} ${1}_${2}${ino} ${2} T “Board ${2}${ino}  Type ${3}” TYPE ${3} NR ${ino}
    ino=‘expr $ino + 1‘
  done
  obj exe
  echo “ Boards inserted”
fi
```

So the command **`board_ins 1U10LA10 AI DSAI_130 3 9`** creates automatically 7 boardelements numbered 3 - 9 below a control node named #1U10LA10

C.6 Example for automatic channel assignment

This example describes a shellscript file, which can be used for “automatic” linking between signal elements and board elements by use of the EIU.

Notice, that all intermediate results within .dat files can be inspected ocular and edited (sort manually, delete datasets), if the user wants it.

The shellscript works in four steps, which can be executed separately, so that the user can inspect the result after each phase:

- reading not used signals of a certain type from the database
- reading not used channels of a certain type from the database
- assigning channels to signals
- importing the result into the database

All needed files are created dynamically by the shellscript.

For the user all needed/produced files are available on the user directory. Those files, which are needed/produced on the EIU directory are copied automatically by the shellscript.

C.6.1 Reading free signal elements from the database

The shellscriptfile ***i_o*** reads signalelements from the database, which are not linked to channel. As parameters must be given:

- method ***sig***
- Objecttype of signalelement
- Objectid (% as wildcard)

Example:

The command "`i_o sig AIS %1K18%`" may provide following result in file AIS_SIG.dat:

```
1K18S5MF110___ACT
1K18MF132___ACT
1K18MF206___ACT
1K18MF207___ACT
1K18MP131___ACT
1K18MP150___ACT
1K18MP204___ACT
1K18MP403___ACT
1K18MT109___ACT
1K18MT133___ACT
1K18MT151___ACT
1K18MT201___ACT
1K18MT202___ACT
1K18MT203___ACT
1K18MT401___ACT
1K18MT421___ACT
```

C.6.2 Reading free channels from the database

The shellscript `i_o` reads also open 'has_signal_element' links from the database. As parameters must be given:

- method ***chn***
- required Objecttype of signalelement
- Object id of boardelemt (% as wildcard)

Example:

The command `"i_o chn  AIS %17%"` may provide following result in file AIS_CH.dat:

```
NODE17_AI2    6
NODE17_AI2    7
NODE17_AI2    8
NODE17_AI2    9
NODE17_AI2   10
NODE17_AI2   11
NODE17_AI2   12
NODE17_AI2   13
```

C.6.3 Assigning signals to free channels

The shellscriptfile i_o merges also the free signals and the free channels together. As parameter only must be given the objecttype.

Example:

The command ***“i_o ass AIS***” using the example data above will provide following result in file
AIS_ASS.dat:

```
NODE17_AI2    6   1K18S5MF110___ACT
NODE17_AI2    7   1K18MF132___ACT
NODE17_AI2    8   1K18MF206___ACT
NODE17_AIS    9   1K18MF207___ACT
NODE17_AI2   10   1K18MP131___ACT
NODE17_AI2   11   1K18MP150___ACT
NODE17_AI2   12   1K18MP204___ACT
NODE17_AI2   13   1K18MP403___ACT
```

C.6.4 Importing of the channel assignment

The produced result can easily be imported by the command “**`i_o imp`**”

A complete example in an overordinated shellscript file may look like:

```
. setup_env
i_o sig AIS %K18%
i_o chn AIS %17%
i_o ass AIS
i_o imp AIS
```

C.6.5 Code of shellscript i_o

```
#Help text
if test “$1” != ’sig’ -a “$1” != ’chn’ -a “$1” != ’ass’ -a “$1” != ’imp’
then
  echo  ’Short description of program use ’
  echo  ’___________________________________________________________________’
  echo  ’The program needs the environment variables’
  echo  ’  UID set to user id’
  echo  ’  PWD set to password’
  echo  ’It should be called therefore from an overordinated shellscript ’
  echo  ’which includes or defines this variables’
  echo  ’The program works in four phases’
  echo  ’ Read unused signals:        i_o sig <signal_type> <obj_id pattern>’
  echo  ’ Read unused channels:       i_o chn <signal_type> <board_id pattern>’
  echo  ’ Assign signals to channels: i_o ass <signal_type>’
  echo  ’ Import assigned signals:    i_o imp <signal_type>’
  echo  ’’
  echo  ’Example’
  echo  ’   ’i_o sig AIS =A10%
  echo  ’   ’i_o chn AIS %N10%
  echo  ’   ’i_o ass AIS
  echo  ’   ’i_o imp AIS
  echo  ’’
fi
# Reading not assigned signals from database
if test “$1” = ’sig’
then
```

```
  eiu export uid=${UID} pwd=${UID} dat=${UID}${2}_SIG.dat tbl=$2 atn=”obj_id” wre=”obj_id LIKE
’${3}’  and nr is null” fmt=fixed
  cp $C_PROJ_EIU/${UID}${2}_SIG.dat ${2}_SIG.dat
  rm $C_PROJ_EIU/${UID}${2}_SIG.dat
  more $C_PROJ_EIU/${UID}${2}_SIG.dat.log
  rm $C_PROJ_EIU/${UID}${2}_SIG.dat.log
  echo File ${2}_SIG.dat create
fi
#
#Reading not used channels from database
if test “$1” = ’chn’
then
  eiu export uid=${UID} pwd=${UID} dat=${UID}${2}_CH.dat tbl=has_signal_element atn=”ob
j_id1,log_channel_no” wre=”obj_id1 LIKE ’${3}’  and type_code=’${2}’and obj_id2 is null”
fmt=fixed
  cp $C_PROJ_EIU/${UID}${2}_CH.dat ${2}_CH.dat
  rm $C_PROJ_EIU/${UID}${2}_CH.dat
  more $C_PROJ_EIU/${UID}${2}_CH.dat.log
  rm $C_PROJ_EIU/${UID}${2}_CH.dat.log
  #sort -o${2}_CH.dat -k 0,1 -k 1n,2n ${2}_CH.dat   #For HP/UX 8.0
  sort -o${2}_CH.dat -k 1,1 -k 2n,3n ${2}_CH.dat    #For HP/UX 9.0
  echo File ${2}_CH.dat created
fi
#
#Assigning Signals to channels
if test “$1” = ’ass’
then
  paste ${2}_CH.dat ${2}_SIG.dat |
  awk ’ NF ==3 {printf “%-32s %2s %-24s \n”, $1, $2, $3}’ > ${2}_ASS.dat
  echo File ${2}_ASS.dat created
fi

#Importing Signal assignment into database
if test “$1” = ’imp’
then
 #Creating the temporary .ctl file
 echo ’LOAD DATA’                                >${C_PROJ_EIU}/ch_imp.ctl
 echo ’INTO TABLE close_temp’                   >>${C_PROJ_EIU}/ch_imp.ctl
 echo ’FIELDS’                                  >>${C_PROJ_EIU}/ch_imp.ctl
 echo ’(’                                       >>${C_PROJ_EIU}/ch_imp.ctl
 echo ’ obj_id1 POSITION (1:32)  CHAR(32),’     >>${C_PROJ_EIU}/ch_imp.ctl
 echo ’ channel POSITION (34:35) CHAR(2),’      >>${C_PROJ_EIU}/ch_imp.ctl
 echo ’ obj_id2 POSITION (37:60) CHAR(24)’      >>${C_PROJ_EIU}/ch_imp.ctl
 echo ’)’                                       >>${C_PROJ_EIU}/ch_imp.ctl
 #Creating the temporary .sel file
 echo ’CATEGORY LINKS’                                                   >${C_PROJ_EIU}/ch_imp.sel
 echo ’TASK CLOSE’                                                      >>${C_PROJ_EIU}/ch_imp.sel
 echo ’SELECT’                                                          >>${C_PROJ_EIU}/ch_imp.sel
 echo ’ATTRIBUTE obj_id1 close_temp.obj_id1’                            >>${C_PROJ_EIU}/ch_imp.sel
 echo  ATTRIBUTE link_type “’”HAS_SIGNAL_ELEMENT”’”                     >>${C_PROJ_EIU}/ch_imp.sel
 echo ’ATTRIBUTE obj_id2 close_temp.obj_id2’                            >>${C_PROJ_EIU}/ch_imp.sel
 echo ’ATTRIBUTE c_int_key has_signal_element.c_int_key’                >>${C_PROJ_EIU}/ch_imp.sel
 echo ’FROM has_signal_element, close_temp’                             >>${C_PROJ_EIU}/ch_imp.sel
 echo ’WHERE has_signal_element.obj_id1 = close_temp.obj_id1 and’       >>${C_PROJ_EIU}/ch_imp.sel
 echo ’      has_signal_element.log_channel_no = close_temp.channel’ >>${C_PROJ_EIU}/ch_imp.sel
 #Starting Import
 cp ${2}_ASS.dat ${C_PROJ_EIU}/${UID}${2}_ASS.dat
 eiu import uid=${UID} pwd=${PWD} sel=ch_imp.sel ctl=ch_imp.ctl dat=${UID}${2}_ASS.dat
 rm $C_PROJ_EIU/${UID}${2}_ASS.dat
 cp $C_PROJ_EIU/${UID}${2}_ASS.dat.log ${2}_ASS.dat.log
 rm $C_PROJ_EIU/${UID}${2}_ASS.dat.log
 more eiu.log
 echo File ${2}_ASS.dat import finished
fi
```

Appendix D Examples for EIU

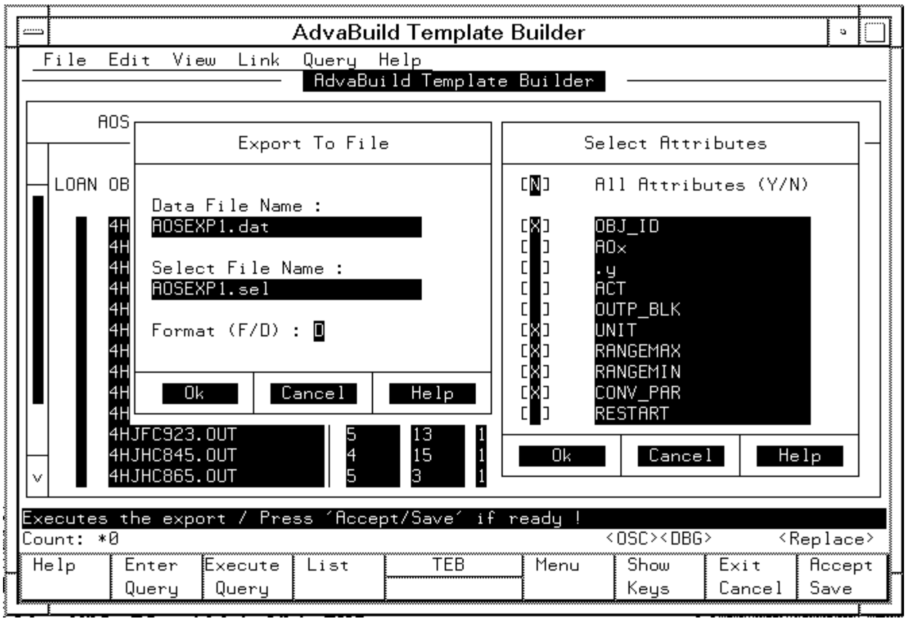
D.1 EIU Export

The given examples are based on the MAS_EXP project

D.1.1 Simple Export with TEB of some Data from one Table

As example we want all AOS Signals with their OBJ_ID, UNIT, RANGEMAX, RANGEMIN and CONV_PAR delimited by comma in ascii format for later use with an other system (e.g. EXEL).

1. Select the **PROJECT** object and choose **Open_Type...** menu from **Object** menu in STB.
2. Enter **AOS** in Inputfield **Selected type** Field and click on **<OK>** button for start TEB.
3. Press **<F5>** key and choose **Export** from **File** menu and then submenu **User Defined...** .
4. Enter the same inputs as displayed in hardcopy, see Figure D-1. Then press <F8>.



Export To File

Data File Name : AOSEXP1.dat

Select File Name : AOSEXP1.sel

Format (F/D) : D

Select Attributes

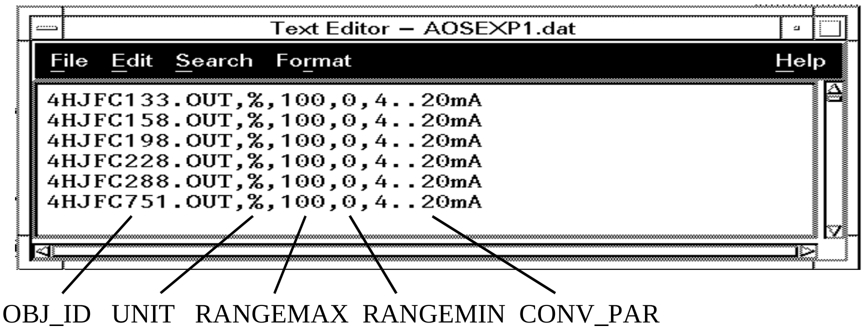
[N] All Attributes (Y/N)

[X] OBJ_ID
[ ] AOx
[ ] .y
[ ] ACT
[ ] OUTP_BLK
[X] UNIT
[X] RANGEMAX
[X] RANGEMIN
[X] CONV_PAR
[ ] RESTART

Executes the export / Press ´Accept/Save´ if ready !
Count: *0 <OSC><DBG> <Replace>

Figure D-1. TEB with EIU parameters

5. The resulting Data File is /proj/mas_exp/eiu/AISEXP1.dat, see Figure D-2



```
4HJFC133.OUT,%,100,0,4..20mA
4HJFC158.OUT,%,100,0,4..20mA
4HJFC198.OUT,%,100,0,4..20mA
4HJFC228.OUT,%,100,0,4..20mA
4HJFC288.OUT,%,100,0,4..20mA
4HJFC751.OUT,%,100,0,4..20mA
```

Figure D-2. Result of Data File

D.1.2 Simple Export with the Command Line of some Data from one Table

As example we want to use the same task as in Chapter D.1.1, Examples for EIU but for export we use the command line instead of the TEB.

1. Change to EIU directory in a hpterm:
   ```
   cd /proj/mas_exp/eiu
   ```
2. Set the environment variable C_PROJ_EIU, enter:
   ```
   export C_PROJ_EIU=/proj/mas_exp/eiu
   ```
3. Start EIU with command line:
   ```
   eiu export uid=mas_exp_cfg pwd=mas_exp_cfg dat=AOSEXP2.dat \
   tbl=AOS atn=OBJ_ID,UNIT,RANGEMAX,RANGEMIN,CONV_PAR \
   wre=1=1 fmt=DELIMITED
   ```

NOTE

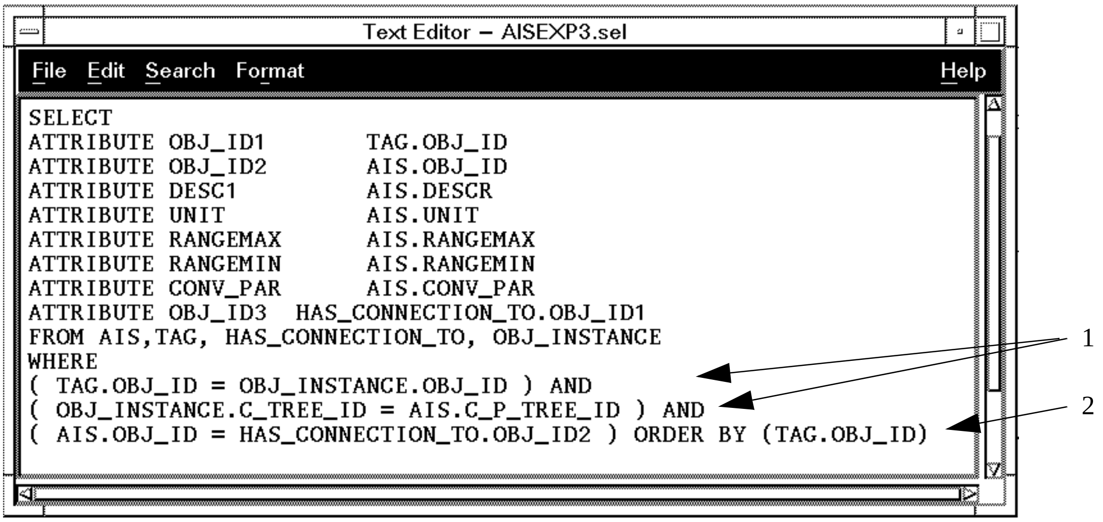
The sign "\" is used as information for the shell, that the line is not ended and the command will be go on the next line. This must be used if the input is to long by entering the data.

4. The file /proj/mas_exp/eiu/AOSEXP2.dat is the same result as file AISEXP1.dat.

D.1.3 Export of Data from Different Tables with Select File

As example we want export all TAGs with their AIS Signals and their OBJ_ID and DESC of the Signals and the used PC to which the Signals are connected.

1. Change to EIU directory in a hpterm:
   ```
   cd /proj/mas_exp/eiu
   ```
2. Use there the preconfigured file AISEXP3.sel, see Figure D-3.



```
SELECT
ATTRIBUTE OBJ_ID1       TAG.OBJ_ID
ATTRIBUTE OBJ_ID2       AIS.OBJ_ID
ATTRIBUTE DESC1         AIS.DESCR
ATTRIBUTE UNIT          AIS.UNIT
ATTRIBUTE RANGEMAX      AIS.RANGEMAX
ATTRIBUTE RANGEMIN      AIS.RANGEMIN
ATTRIBUTE CONV_PAR      AIS.CONV_PAR
ATTRIBUTE OBJ_ID3  HAS_CONNECTION_TO.OBJ_ID1
FROM AIS,TAG, HAS_CONNECTION_TO, OBJ_INSTANCE
WHERE
( TAG.OBJ_ID = OBJ_INSTANCE.OBJ_ID ) AND
( OBJ_INSTANCE.C_TREE_ID = AIS.C_P_TREE_ID ) AND
( AIS.OBJ_ID = HAS_CONNECTION_TO.OBJ_ID2 ) ORDER BY (TAG.OBJ_ID)
```

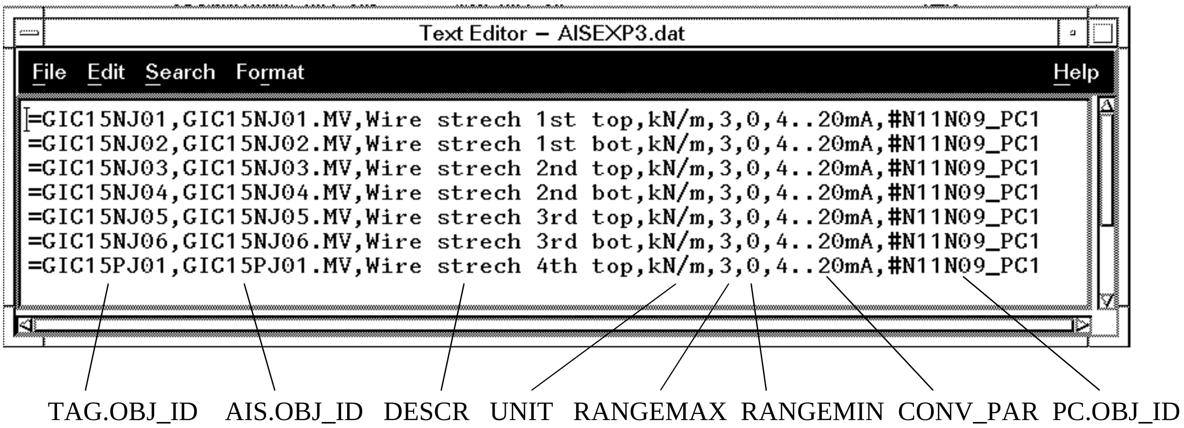
Figure D-3. Select File for EIU Export

The relation between TAG and Signal is, that the TAG is the parent of the Signal (see WHERE condition 1).
The relation between Signal and PC is the connection with HAS_CONNECTION_TO Link (see WHERE condition 2).

NOTE

By select from tables, which have same Attributes you must add a mark in front of the Attribute name. In this example we used the table names (e.g. AIS -> AIS.OBJ_ID) as mark. For other possible marks like alias names from tables see SQL language description.

3. Set the environment variable C_PROJ_EIU, enter:
   export C_PROJ_EIU=/proj/mas_exp/eiu
4. Start EIU with command line:
   ```
   eiu export uid=mas_exp_cfg pwd=mas_exp_cfg dat=AOSEXP3.dat \
   sel=AISEXP3.sel
   ```
5. The resulting Data File is /proj/mas_exp/eiu/AISEXP3.dat,Figure D-4.



```
=GIC15NJ01,GIC15NJ01.MV,Wire strech 1st top,kN/m,3,0,4..20mA,#N11N09_PC1
=GIC15NJ02,GIC15NJ02.MV,Wire strech 1st bot,kN/m,3,0,4..20mA,#N11N09_PC1
=GIC15NJ03,GIC15NJ03.MV,Wire strech 2nd top,kN/m,3,0,4..20mA,#N11N09_PC1
=GIC15NJ04,GIC15NJ04.MV,Wire strech 2nd bot,kN/m,3,0,4..20mA,#N11N09_PC1
=GIC15NJ05,GIC15NJ05.MV,Wire strech 3rd top,kN/m,3,0,4..20mA,#N11N09_PC1
=GIC15NJ06,GIC15NJ06.MV,Wire strech 3rd bot,kN/m,3,0,4..20mA,#N11N09_PC1
=GIC15PJ01,GIC15PJ01.MV,Wire strech 4th top,kN/m,3,0,4..20mA,#N11N09_PC1
```

Figure D-4. Result of Data File

D.1.4 Extract Files (Texts) from DB with EIU

As example we want extract all Text files from the database to filesystem, so that you can transfer later the files to other platforms.
As first we will create with EIU a Text-Object list, which we use later for extract with EIU as input list.

1. Change to EIU directory in a hpterm:
   cd /proj/mas_exp/eiu
2. Set the environment variable C_PROJ_EIU (this ia also the directory, in which EIU will later extract the files), enter:
   export C_PROJ_EIU=/proj/mas_exp/eiu
3. Create with EIU a Text-Object list with file-name for the Text-files.
   Start EIU with command line:

```
eiu export uid=mas_exp_cfg pwd=mas_exp_cfg dat=TEXT.dat \
tbl=TEXT atn=OBJ_ID wre=1=1 fmt=DELIMITED
```

This creates an TEXT.dat file with a list of all objects with type TEXT.

4. Use now the predefined files TEXTEXP.ctl, TEXTEXP.sel and your new created TEXT.dat in directory /proj/mas_exp/eiu. Start EIU with command line:

```
eiu export uid=mas_exp_cfg pwd=mas_exp_cfg dat=TEXT.dat \
sel=TEXTEXP.sel ctl=TEXTEXP.ctl eop=CR
```

File TEXTEXP.ctl:

```
LOAD DATA
INTO TABLE TMPIMP1
FIELDS TERMINATED BY ‘,’
(
OBJ_ID          char(20)
)
```

File TEXTEXP.sel:

```
TASK EXTRACT
SELECT
ATTRIBUTE OBJ_ID      OBJ_ID
ATTRIBUTE FILE_NAME   OBJ_ID || ‘.frm’
FROM TMPIMP1
WHERE 1=1
```

Remark:
The TEXT.dat file is the file for the extract filled with all TEXT objects. The TEXTEXP.ctl describe the format of TEXT.dat for the extract. In TEXTEXP.sel file all objects from TEXT.dat file will be selected and extracted from DB to FrameMaker files with object name + ‘.frm’ als extension for filename. The files will be written to the predefined environment variable C_PROJ_EIU (/proj/mas_exp/eiu).

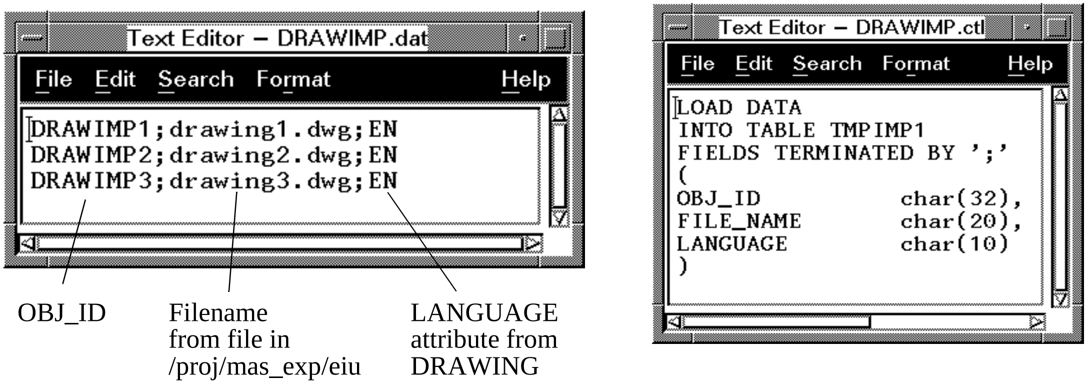
D.2 EIU Import

The given examples are based on ;the MAS_EXP project

D.2.1 Import Files (Drawings) in DB with EIU

As example we want import three drawings from directory /proj/mas_exp/eiu with the names drawing1.dwg to drawing3.dwg. These drawings are dwg files and was created by AUTOCAD on a other platform. The files was created in AUTOCAD with the wblock command, because with the normal save command the dwg- files get the environment from the other platform, which can be different.

As input for the Data File we use the predefined DRAWIMP.dat file, with semi-colon as delimiter, and the corresponding DRAWIMP.ctl as Control File, see Figure D-5.


Text Editor – DRAWIMP.dat
File Edit Search Format Help
Text Editor – DRAWIMP.ctl
File Edit Search Format Help

```
DRAWIMP1;drawing1.dwg;EN
DRAWIMP2;drawing2.dwg;EN
DRAWIMP3;drawing3.dwg;EN
```

OBJ_ID

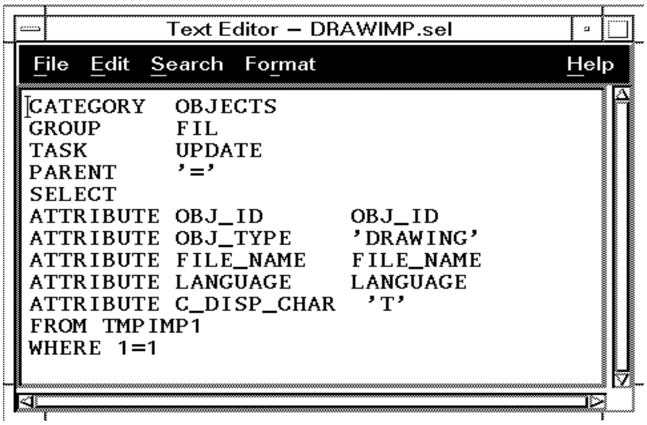
Filename
from file in
/proj/mas_exp/eiu

LANGUAGE
attribute from
DRAWING

```
LOAD DATA
INTO TABLE TMPIMP1
FIELDS TERMINATED BY ';'
(
OBJ_ID           char(32),
FILE_NAME        char(20),
LANGUAGE         char(10)
)
```

Figure D-5. Data and Control File

The Select File describe, that the drawings will be inserted as DRAWING objects in Tree Representation and the defined values from Data File will be used for OBK_ID, FILE_NAME and LANGUAGE, see Figure D-6.



```
CATEGORY  OBJECTS
GROUP     FIL
TASK      UPDATE
PARENT    '='
SELECT
ATTRIBUTE OBJ_ID       OBJ_ID
ATTRIBUTE OBJ_TYPE     'DRAWING'
ATTRIBUTE FILE_NAME    FILE_NAME
ATTRIBUTE LANGUAGE     LANGUAGE
ATTRIBUTE C_DISP_CHAR   'T'
FROM TMPIMP1
WHERE 1=1
```

Figure D-6. Select File

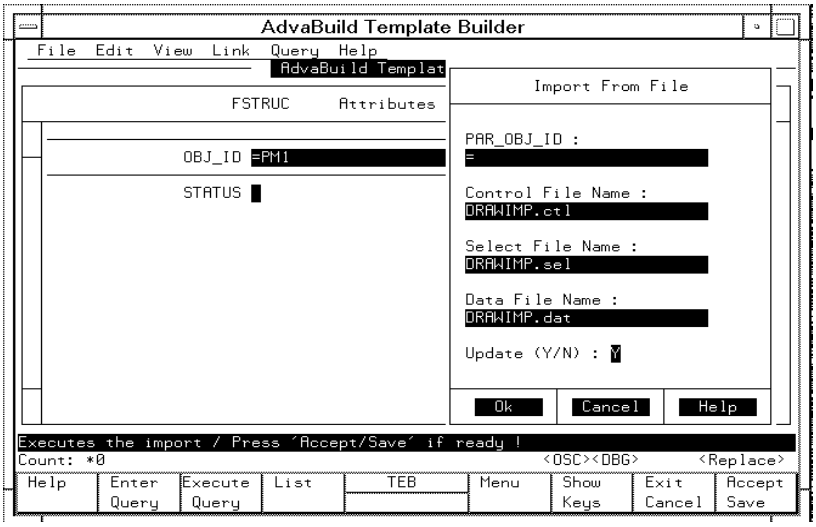
1. Change to EIU directory in a hpterm:
   cd /proj/mas_exp/eiu
2. Set the environment variable C_PROJ_EIU (this ia also the directory, in which EIU will search for the drawing files, if not attribute PATH in select file is used), enter:
   export C_PROJ_EIU=/proj/mas_exp/eiu
3. Start EIU with command line:

```
eiu import uid=mas_exp_cfg pwd=mas_exp_cfg dat=DRAWIMP.dat \
ctl=DRAWIMP.ctl sel=DRAWIMP.sel
```

Or:

1. Select object "=PM1" and choose **Open** menu from **Object** menu in STB.
2. Press **<F5>** key and choose **Import...** from **File** menu.

3. Enter the same inputs as displayed in hardcopy, see Figure D-7. Then press <F8>.



Import From File

PAR_OBJ_ID : =

Control File Name : DRAWIMP.ctl

Select File Name : DRAWIMP.sel

Data File Name : DRAWIMP.dat

Update (Y/N) : Y

OBJ_ID =PM1

STATUS

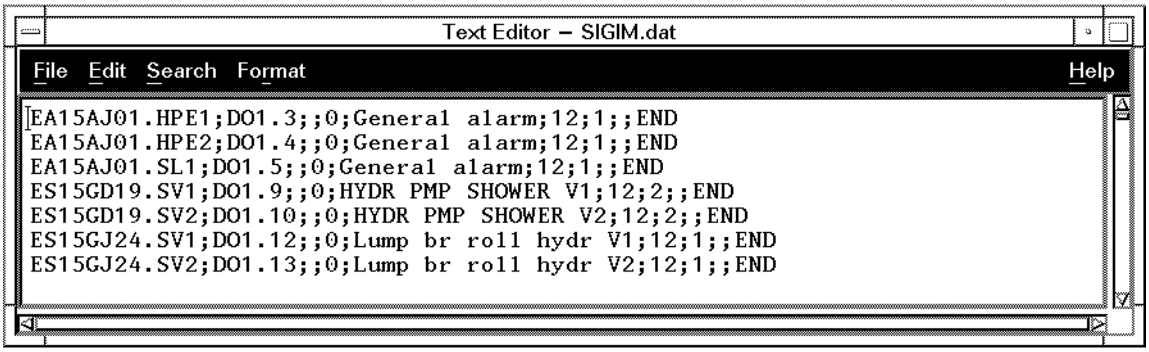
Executes the import / Press ´Accept/Save´ if ready !

Count: *0

Figure D-7. TEB with EIU parameters

D.2.2 Import of Digital Signals with Structure Objects, Boards and Links

As input for the data we use the predefined SIGIM.dat file, which was produced by a EXEL export with semi-colon as delimiter, see Figure D-8. These data should be inserted in the AS500ES object structure to generate Source Code for a MP200.

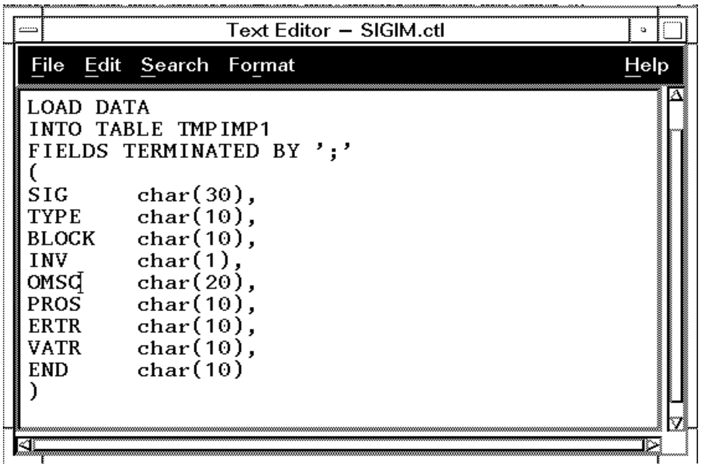


```
EA15AJ01.HPE1;DO1.3;;0;General alarm;12;1;;END
EA15AJ01.HPE2;DO1.4;;0;General alarm;12;1;;END
EA15AJ01.SL1;DO1.5;;0;General alarm;12;1;;END
ES15GD19.SV1;DO1.9;;0;HYDR PMP SHOWER V1;12;2;;END
ES15GD19.SV2;DO1.10;;0;HYDR PMP SHOWER V2;12;2;;END
ES15GJ24.SV1;DO1.12;;0;Lump br roll hydr V1;12;1;;END
ES15GJ24.SV2;DO1.13;;0;Lump br roll hydr V2;12;1;;END
```

Figure D-8. SIGIM.dat

The first field is the signal name, also the Functional Group name (3 characters beginning from character "1") and the TAG name (characters until character "."). For Functional Group name and TAG name also the STRUC name is the beginning Step of the name.
The second field represents the Signal Type (characters until character "."), the Board name and (characters until character ".") and the Link information, to which Board and Channel No. the Signal is linked to.
The rest of the fields:
3 field is for attribute BLOCK (only for DIS signals)
4 field is for attribute INV Inverse switch)
5 field is for attribute SYM_NAME (description for symbolic name)
6 field is for attribute ERR_TR
7 field is for attribute VALUE_TR (only for DIS signals)
8 field is only a dummy, that the last row is not null for better use of Oracle7

The mapping to the Temporary Table from SIGIM.dat file you can see in the predefined SIGIM.ctl file.



```
LOAD DATA
INTO TABLE TMPIMP1
FIELDS TERMINATED BY ';'
(
SIG      char(30),
TYPE     char(10),
BLOCK    char(10),
INV      char(1),
OMSC     char(20),
PROS     char(10),
ERTR     char(10),
VATR     char(10),
END      char(10)
)
```

Figure D-9. SIGIM.dat

For import use the predefined SIGIM.sel file. This file is sorted in 7 Steps. Each Step is also be able to run alone separately in a selectfile, if before the other relevant Steps was run.

NOTE

In these Steps were used a WHERE condition with table ROWID. This is only a small trick to avoid multi insert of same hidden selects in where condition and so resulting errors by inserting a already inserted object in Log-file.
The SQL-Syntax: '<string1>' || '<srting2>'
will connect these to strings to one new string.
The SQL-Syntax: SUBSTR (<string1>,m,[n])
will create a substring of <string1>, beginning at position m, [n]positions long (if [n] is not omitted, to end of <string1>).
The SQL-Syntax: INSTR (<string1>,<char1>,[n],[m])
will get the position of the [m]th occurrence of <char1> in <string1>, beginning search at position [n] (if [n] or [m] is not set, default 1 is used).

1 Step:	Create the FSTRUCs, if not already exist, under object "=PM1DRY".
2 Step:	Create the FGRPs if not already exist under created FSTRUC objects.
3 Step:	Create the TAGs if not already exist under created FGRP objects.
4 Step:	Create the DIS Signals if not already exist under TAG objects.
5 Step:	Create the DOS Signals if not already exist under TAG objects.
6 Step:	Create the DI and DO Boards, if not already exist, under object "#N11N09".
7 Step:	Create Links to the empty Channels of the created Boards.

File SIGIM.sel:

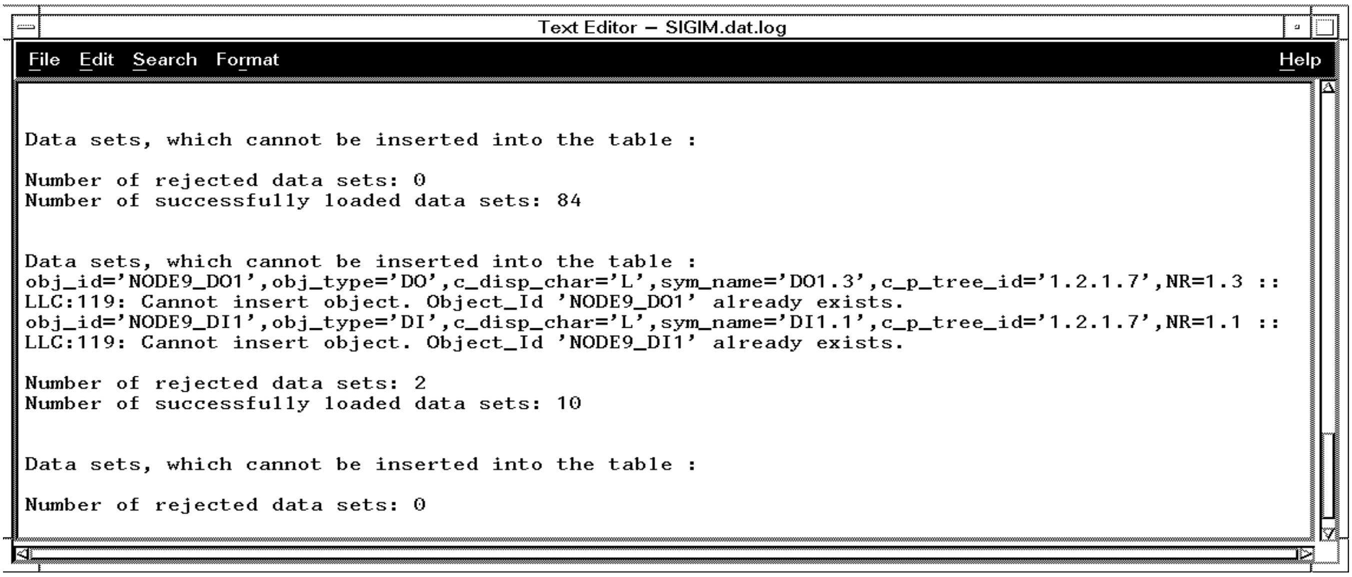
```
--1 Step
CATEGORY OBJECTS
TASK     INSERT
```

```
PARENT    ‘=PM1DRY’
SELECT
ATTRIBUTE OBJ_ID       ‘=PM1DRY’ || SUBSTR(SIG,INSTR(SIG,’1’,1,1),3)
ATTRIBUTE OBJ_TYPE     ‘FSTRUC’
ATTRIBUTE SYM_NAME     ‘New F. structure EIU’
ATTRIBUTE C_DISP_CHAR  ‘T’
FROM             TMPIMP1
WHERE            ROWID IN
                 (SELECT MIN(DISTINCT ROWID) FROM TMPIMP1
                 GROUP BY SUBSTR(SIG,INSTR(SIG,’1’,1,1),3))

--2 Step
CATEGORY  OBJECTS
TASK      INSERT
PARENT    ‘=PM1DRY’ || SUBSTR(SIG,INSTR(SIG,’1’,1,1),3)
SELECT
ATTRIBUTE OBJ_ID       ‘=PM1DRY’ || SUBSTR(SIG,INSTR(SIG,’1’,1,1),4)
ATTRIBUTE OBJ_TYPE     ‘FGRP’
ATTRIBUTE SYM_NAME     ‘New F. group EIU’
ATTRIBUTE C_DISP_CHAR  ‘T’
FROM             TMPIMP1
WHERE            ROWID IN
                 (SELECT MIN(DISTINCT ROWID) FROM TMPIMP1
                 GROUP BY SUBSTR(SIG,INSTR(SIG,’1’,1,1),4))

--3 Step
CATEGORY  OBJECTS
TASK      INSERT
PARENT    ‘=PM1DRY’ || SUBSTR(SIG,INSTR(SIG,’1’,1,1),4)
SELECT
ATTRIBUTE OBJ_ID       ‘=’ || SUBSTR(SIG,1,INSTR(SIG,’.’,1,1)-1)
ATTRIBUTE OBJ_TYPE     ‘TAG’
ATTRIBUTE SYM_NAME     OMSC
ATTRIBUTE C_DISP_CHAR  ‘T’
FROM             TMPIMP1
WHERE            ROWID IN
                 (SELECT MIN(DISTINCT ROWID) FROM TMPIMP1
                 GROUP BY SUBSTR(SIG,1,INSTR(SIG,’.’,1,1)-1))
--4 Step
CATEGORY  OBJECTS
TASK      INSERT
PARENT    ‘=’ || SUBSTR(SIG,1,INSTR(SIG,’.’,1,1)-1)
SELECT
ATTRIBUTE OBJ_ID       SIG
ATTRIBUTE OBJ_TYPE     ‘DIS’
ATTRIBUTE SYM_NAME     OMSC
ATTRIBUTE BLOCKED      BLOCK
ATTRIBUTE INV          INV
ATTRIBUTE PROC_SEC     PROS
ATTRIBUTE ERR_TR       ERTR
ATTRIBUTE VALUE_TR     VATR
ATTRIBUTE C_DISP_CHAR  ‘T’
FROM               TMPIMP1
WHERE              SUBSTR(TYPE,1,2) = ’DI’

5 Step
CATEGORY  OBJECTS
TASK      INSERT
PARENT    ‘=’ || SUBSTR(SIG,1,INSTR(SIG,’.’,1,1)-1)
SELECT
ATTRIBUTE OBJ_ID       SIG
ATTRIBUTE OBJ_TYPE     ‘DOS’
ATTRIBUTE SYM_NAME     OMSC
ATTRIBUTE INV          INV
ATTRIBUTE PROC_SEC     PROS
ATTRIBUTE ERR_TR       ERTR
ATTRIBUTE C_DISP_CHAR  ‘T’
FROM               TMPIMP1
WHERE              SUBSTR(TYPE,1,2)=’DO’

6 Step
CATEGORY  OBJECTS
TASK      INSERT
PARENT    ‘#N11N09’
SELECT
ATTRIBUTE OBJ_ID       ‘NODE9_’||SUBSTR(TYPE,1,INSTR(TYPE,’.’)-1)
ATTRIBUTE OBJ_TYPE     SUBSTR(TYPE,1,2)
ATTRIBUTE SYM_NAME     SUBSTR(TYPE,1,5)
ATTRIBUTE NR       SUBSTR(TYPE,3,INSTR(TYPE,’.’)-1)
ATTRIBUTE C_DISP_CHAR  ‘L’
FROM               TMPIMP1
WHERE             ROWID IN (SELECT MIN(DISTINCT ROWID) FROM TMPIMP1
                 WHERE SUBSTR(TYPE,1,2) IN (‘DI’,’DO’)
                  GROUP BY SUBSTR(TYPE,1,INSTR(TYPE,’.’)-1))



--7 Step
CATEGORY  LINKS
TASK      CLOSE
SELECT
ATTRIBUTE OBJ_ID1       ‘NODE9_’ ||
SUBSTR(TMPIMP1.TYPE,1,INSTR(TMPIMP1.TYPE,’.’)-1)
ATTRIBUTE LINK_TYPE  ‘HAS_SIGNAL_ELEMENT’
ATTRIBUTE OBJ_ID2         SIG
ATTRIBUTE C_INT_KEY     HAS_SIGNAL_ELEMENT.C_INT_KEY
FROM      TMPIMP1, HAS_SIGNAL_ELEMENT
WHERE     HAS_SIGNAL_ELEMENT.OBJ_ID1 =
                    ‘NODE9_’ || SUBSTR(TMPIMP1.TYPE,1,INSTR(TMPIMP1.TYPE,’.’)-1)
```

```
AND HAS_SIGNAL_ELEMENT.LOG_CHANNEL_NO =
                            SUBSTR(TMPIMP1.TYPE,INSTR(TMPIMP1.TYPE,'.')+1,2)
AND HAS_SIGNAL_ELEMENT.OBJ_ID2 IS NULL
```

Now you can start the Import:

1. Change to EIU directory in a hpterm:
   cd /proj/mas_exp/eiu
2. Set the environment variable C_PROJ_EIU (this ia also the directory, in which EIU will later extract the files), enter:
   export C_PROJ_EIU=/proj/mas_exp/eiu
3. Start EIU with command line:
   ```
   eiu import uid=mas_exp_cfg pwd=mas_exp_cfg dat=SIGIM.dat \
   ctl=SIGIM.ctl sel=SIGIM.sel
   ```
4. Accept the Data Error Message by pressing <OK> button and open the Logfile with vuepad, see Figure D-10:
   ```
   vuepad SIGIM.dat.log &
   ```



```
Data sets, which cannot be inserted into the table :

Number of rejected data sets: 0
Number of successfully loaded data sets: 84


Data sets, which cannot be inserted into the table :
obj_id='NODE9_DO1',obj_type='DO',c_disp_char='L',sym_name='DO1.3',c_p_tree_id='1.2.1.7',NR=1.3 ::
LLC:119: Cannot insert object. Object_Id 'NODE9_DO1' already exists.
obj_id='NODE9_DI1',obj_type='DI',c_disp_char='L',sym_name='DI1.1',c_p_tree_id='1.2.1.7',NR=1.1 ::
LLC:119: Cannot insert object. Object_Id 'NODE9_DI1' already exists.

Number of rejected data sets: 2
Number of successfully loaded data sets: 10


Data sets, which cannot be inserted into the table :

Number of rejected data sets: 0
```

Figure D-10. Logfile

Here you can see an example, that objects "NODE9_DO1" and "NODE9_DI1" cannot inserted in the project. The reason is that these objects already exist so that they can not be inserted twice.

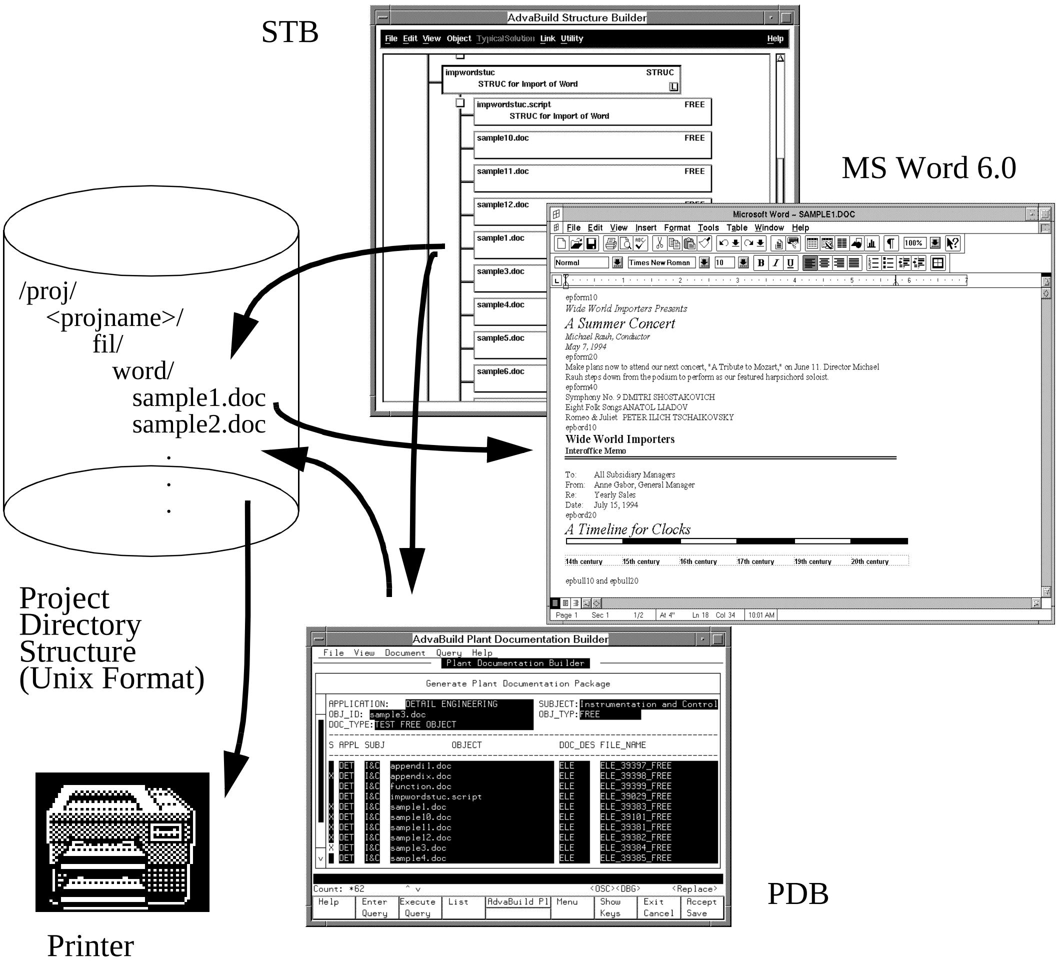
5. Now you are able to start FCB and create Source Code with the new Boards and Signals on Node "#N11N09

Appendix E Use of Microsoft Word 6.0 documents in AS500ES

E.1 Introduction

This section describe how to use Microsoft Word documents in AS500ES. Preconditions for use of Microsoft Word on AS500ES is a software emulator for use of Microsoft Word based on Microsoft Windows for HP-UNIX.
This description is written for use software emulator Wabi 2.0 for HP-UX 9.05 with Microsoft Windows 3.1 and Microsoft Word 6.0.

The documents are saved on the filesystem and have FREE objects as represents in STB for open, print and PDB print method (see Figure E-1).



epform10
Wide World Importers Presents
A Summer Concert
Michael Rauh, Conductor
May 7, 1994
epform20
Make plans now to attend our next concert, "A Tribute to Mozart," on June 11. Director Michael Rauh steps down from the podium to perform as our featured harpsichord soloist.
epform40
Symphony No. 9 DMITRI SHOSTAKOVICH
Eight Folk Songs ANATOL LIADOV
Romeo & Juliet PETER ILICH TSCHAIKOVSKY
epbord10
Wide World Importers
Interoffice Memo

To: All Subsidiary Managers
From: Anne Gabor, General Manager
Re: Yearly Sales
Date: July 15, 1994
epbord20
A Timeline for Clocks
14th century 15th century 16th century 17th century 19th century 20th century

epbull10 and epbull20

Page 1 Sec 1 1/2 At 4" Ln 18 Col 34 10:01 AM

APPLICATION: DETAIL ENGINEERING SUBJECT: Instrumentation and Control
OBJ_ID: sample3.doc OBJ_TYP: FREE
DOC_TYPE: TEST FREE OBJECT

S	APPL	SUBJ	OBJECT	DOC_DES	FILE_NAME
	DET	I&C	appendi1.doc	ELE	ELE_39397_FREE
X	DET	I&C	appendix.doc	ELE	ELE_39398_FREE
	DET	I&C	function.doc	ELE	ELE_39399_FREE
	DET	I&C	impwordstuc.script	ELE	ELE_39029_FREE
X	DET	I&C	sample1.doc	ELE	ELE_39383_FREE
X	DET	I&C	sample10.doc	ELE	ELE_39101_FREE
X	DET	I&C	sample11.doc	ELE	ELE_39381_FREE
X	DET	I&C	sample12.doc	ELE	ELE_39382_FREE
X	DET	I&C	sample3.doc	ELE	ELE_39384_FREE
	DET	I&C	sample4.doc	ELE	ELE_39385_FREE

Count: *62 ^ v <OSC><DBG> <Replace>

Figure E-1. Free objects in STB

E.2 How to establish Wabi, Microsoft Windows and Microsoft Word

E.2.1 Installation of Software for Use of Microsoft Word

- After finishing the installation of a AS500ES (with release equal or higher rel.2.5), install as first step Wabi as described in your Wabi documentation.
  For wabi installation directory use:
  `/opt/wabi`
  For Wabi-User (not DB user) directory use:
  `/users/<user>/wabi`
  (e.g. for <user> is master1: `/users/master1/wabi`)
  Remark: You must install wabi for the HP-user, who is used later for work with STB and Wabi in project.
- As next install Windows3.1 as described in Wabi and Windows documentation.
  For Windows3.1-Wabi-User directory use:
  `/users/<user>/wabi/windows`
- As next install Microsoft Word 6.0 as described in Word documentation.
  For Microsoft Word 6.0-Wabi-User directory use:
  `/users/<user>/wabi/msoffice/winword`

  Remark: These directories must be used, other directories will not be supported for communication between STB and Microsoft Word.

E.2.2 Software Configuration for Use in AS500ES

Wabi and Mircrosoft Windows 3.1 relevant changes in configuration for use in AS500ES

- For Wabi Drive Connections, you must connect in Wabi Configuration:
  Drive `C:` to `$HOME/wabi`
  Drive `R:` to `/`
  Remark: $HOME is an environment variable for /users/<user> directory.
- For Wabi Printer Settings, you must set in Wabi Configuration:
  Default Printer to Port (Port Selection):
  `LPT1:`
  connected to printer queue (in Port Selection by Native Printer Name):
  `lpraw`
  with Native Print Command:
  `lp -oraw -d%p -t%t`
  Check also the correct Paper Size format (A4 210x297mm or Letter 81/2x11) in Wabi Printer Settings. If this is not correct set, the printer will stop and ask you by every printout of a page.

Microsoft Word 6.0 relevant changes in configuration for use in AS500ES

- Check, that for printer the default printer from Windows 3.1 is used.

E.2.3 Check Correct Installation and Configuration

- Start Wabi 2.0 with Windows 3.1 from a HP-term as described in Wabi documentation.
- Open Windows 3.1 Filemanager and check access to connected Drives.
- Start Microsoft Word 6.0 and print a document to default printer.

E.2.4 Microsoft Word Configuration for Communication with STB

For communication between STB and Microsoft Word a Macro is needed. Start Microsoft Word and load text file:

R:\proj\<proj_name>\util\free\autoexec.txt

Select the whole text by use of <Control> + <c>, this save the contents to the buffer. Create a new Macro available for Normal.dot Word-Template with name:

AutoExec

Edit this macro and paste in the saved buffer with <Control>+<v>. Then save the new Macro and exit Microsoft Word. For more information regarding the macro see Microsoft Word manual.

Cleanup `startup` directory for Microsoft Word, else some problems with other macros can arise (Convert dialog by startup).

E.2.5 Creating of Microsoft Word Document Directory

For every project a own directory for use of Microsoft Word 6.0 documents are used.
This directories are located below the `fil` directory with name `word`:

/proj/<proj_name>/fil/word

Create this word directory on UNIX platform, if not existing, and copy your Mircrosoft Word 6.0 documents with extension "`.doc`" to this directory.

E.3 Open, Print in STB from Microsoft Word Documents

E.3.1 Creation of FREE Object for Microsoft Word

For every Mircrosoft Word 6.0 document a FREE object must be as represent. These FREE object must filled with the correct data for the use of Mircrosoft Word 6.0 documents.

OBJECT_ID:

The OBJECT_ID of the FREE object is defined through Mircrosoft Word file_name without extension "`.doc`".

E.g.: OBJECT_ID is "`sample1.doc`" will be a represent for file with name "`sample1.doc`" in `/proj/<proj_name>/fil/word` directory.

Remark: Use only 8 characters in front of "`.doc`" by OBJECT_ID, as defined for DOS/Windows convention for filesystems. Also only small characters are allowed for the OBJECT_ID, because wabi will use only small characters.

ATTRIBUTES:

- SCRIPT_FILE must be filled with:
  `wabiword.sh`
- DESCR_1 only description:
  `Command for Wabi`
- PARAM_1 must be filled with shell command for start of Wabi including the path in UNIX format:
  `/opt/wabi/bin/wabi`
- DESCR_2 only description:
  `Wabi User`
- PARAM_2 must be filled with configured Wabi user e.g. for master user:
  `master`
  or user variable:
  `$USER`

For import of already created Word documents, see Chapter E.3.5, Use of Microsoft Word 6.0 documents in AS500ES

E.3.2 STB-Open of Microsoft Word Document with FREE Object as Representatives

- Double Click on the established FREE object or select FREE object and select in STB menu <Object> + <Open...>.
- Select in appearing Open Object window the view FREE and press <OK> button.

Now Microsoft Word 6.0 will be started with Wabi. If OBJECT_ID and the document file exist in /proj/<proj_name>/fil/word directory, the file will be opened and used as document.

If the document file in /proj/<proj_name>/fil/word directory is not existing, it will be created automatically as empty document and opened with Microsoft Word 6.0.

E.3.3 STB-Print of Microsoft Word Document with FREE Object as Representative

- Select FREE object and select in STB menu <Object> + <Print...>.
- Select in appearing Print Object window the view FREE and press <OK> button for direct printing.
- Or select in appearing Print Object window the view FREE and push-button 'Print only to file' and enter a file name for printing to filesystem. Than the file will be saved with file_name plus extension '.prn' on directory /proj/<proj_name>/usr/<db_user_name> and can print from a hpterm on lprow queue.

E.3.4 PDB Configuration for Microsoft Word 6.0 Documents

Before you can make printouts with PDB-Print you must configure your PDB for Microsoft Word document with FREE object as represent.

- Select in STB menu <Utility> + <Configure>-<Plant_Documentation Builder>
- Insert a new row as described in PDB manual for a new selection from PDB. For following attributes enter:
- OBJ_TYPE must edit with:
  `FREE`
- COMMAND must edit with:
  `PRINT`
- COMMAND must edit with:
  `FREE`
- W_TIME must edit with largest generation time of the biggest document e.g.:
  150
- CYCLES must edit:
  `2`
- QT must edit with:
  `LR`

Remark: By printout with PDB select only FREE objects with extension in OBJECT ID "`.doc`", because only this represent Microsoft Word documents.

E.3.5 Import of Microsoft Word 6.0 Documents from other Systems

For automatic creation of FREE objects as represents of Microsoft Word 6.0 documents a scripted is designed for a FREE object, which will call the EIU.
This script will search for new Microsoft Word 6.0 documents in path:
`/proj/<proj_name>/fil/word`
with name conventions (8 characters plus extension`.doc` in small characters):
`<filename>.doc`
If new Microsoft Word 6.0 documents are found, the representing FREE objects will be created in STB.
The parent OBJECT_ID and the display in STB can be given by parameters of this FREE object for import of Microsoft Word 6.0 documents.

Create a new FREE object for import Microsoft Word 6.0 documents with name:
`impwordobj.script`
Edit attributes in TEB:

- SCRIPT_FILE must be filled with:
  `impword.sh`
- DESCR_1 only description:
  `Parent object name`
- PARAM_1 must be filled with an existing Parent OBJ_ID of a Tree object e.g.:
  `=PM1`
- DESCR_2 only description:
  `Display of object`
- PARAM_2 must be filled with display of the FREE object (T for Tree, L for LIST)e.g.:
  `L`

Now copy your new Microsoft Word 6.0 documents with extension "`.doc`" from other systems to `directory:/proj/<proj_name>/fil/word`
Then double Click on the established FREE object and execute the import by selecting view FREE and press <OK> button.

Appendix F Use of Time controlled Central Backup with Free Object

F.1 Introduction

This section describes the Time controlled Central Backup with the user interface of FREE objects. Minimal preconditions for the usage of Time controlled Central Backup are that the MAS_EXP Project is installed, which includes the predefined FREE objects and corresponding FREE-Scripts for use. Also the ONB must be installed and configured correctly and RTA-Board must be correctly booted.

F.1.1 Time controlled Central Backup Function in MAS_EXP Project

1. Please login in the AS500ES as one of the predefined master user.
2. Check that ONB is installed and configured correctly and is not in use.
3. Also check that the RTA-Board is correct booted and running.
4. Start the Database and open the STB on project "MAS_EXP".
   Open the Tree-Child-Objects from "PROJECT_LIBRARY" and then from "FREE_LIB".
   Now the "START_CRON" and "STOP_CRON" objects will appear (see Figure F-1).

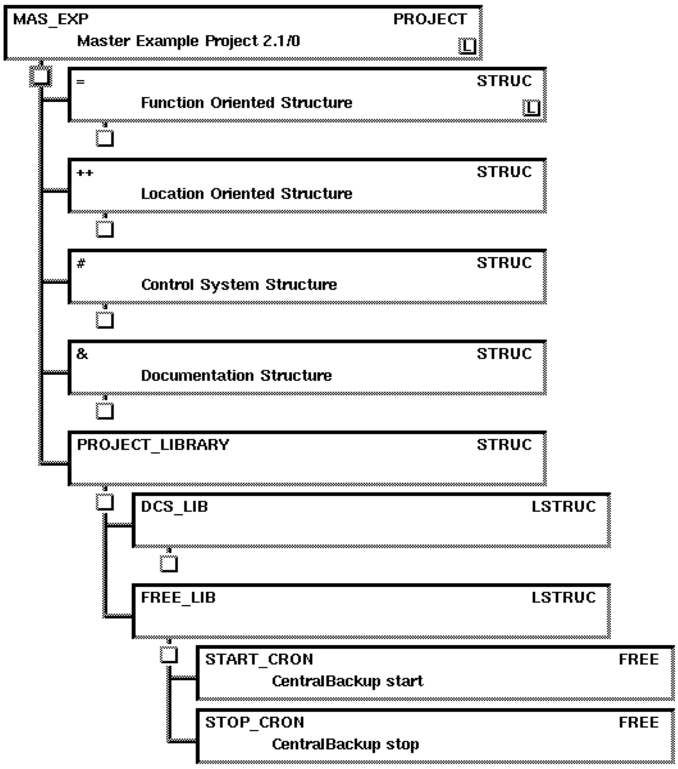


Figure F-1. START_CRON and STOP_CRON Objects as FREE Objects

F.1.1.1 Stop of an already running Time controlled Central Backup

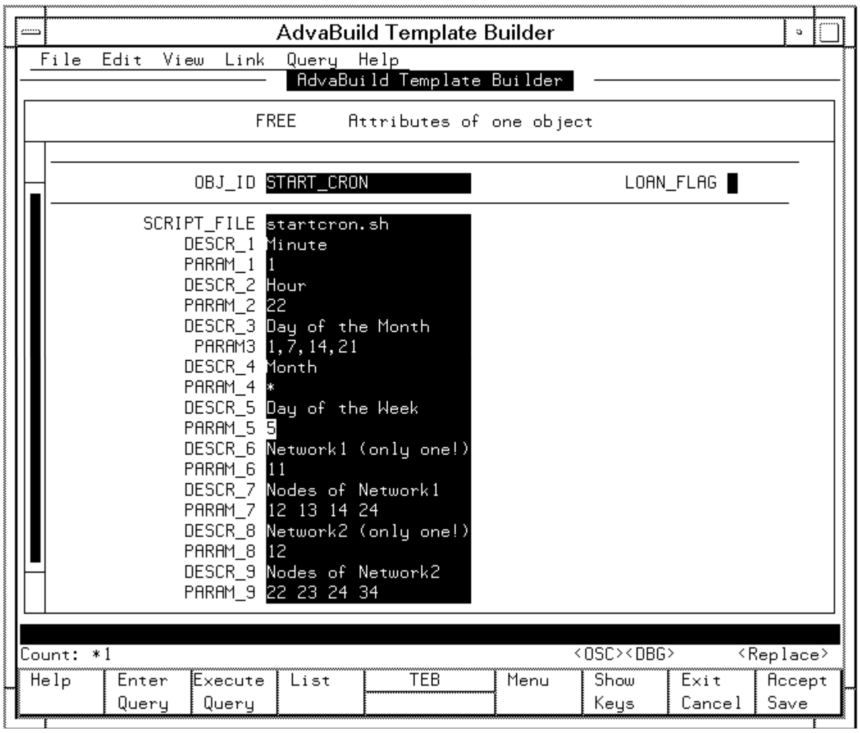
1. Open Object “STOP_CRON”.
2. Select “FREE” in the appearing “Open Object” window and press the “OK” button.

NOTE

This action will remove your entry in the crontab of HP-UX for start-up of the Time controlled Central Backup (do backup script) of the current HP-UX user (login in AS500ES). If you have started the Time controlled Central Backup as an other user before, the process of this user will not be affected.

F.1.1.2 Start of a Time controlled Central Backup

1. Open Object “START_CRON”.
2. Select “TEMPLATE” in appearing “Open Object” window and press “OK” button, now the TEB will be started (see Figure F-2).


AdvaBuild Template Builder
File Edit View Link Query Help
AdvaBuild Template Builder
Help
Enter Query
Execute Query
List
TEB
Menu
Show Keys
Exit Cancel
Accept Save

FREE Attributes of one object

OBJ_ID START_CRON
LOAN_FLAG

SCRIPT_FILE startcron.sh
DESCR_1 Minute
PARAM_1 1
DESCR_2 Hour
PARAM_2 22
DESCR_3 Day of the Month
PARAM3 1,7,14,21
DESCR_4 Month
PARAM_4 *
DESCR_5 Day of the Week
PARAM_5 5
DESCR_6 Network1 (only one!)
PARAM_6 11
DESCR_7 Nodes of Network1
PARAM_7 12 13 14 24
DESCR_8 Network2 (only one!)
PARAM_8 12
DESCR_9 Nodes of Network2
PARAM_9 22 23 24 34

Count: *1 <OSC><DBG> <Replace>

Figure F-2. Parameters of FREE Object START_CRON

3. Enter your correct values for the following attributes (see also Figure F-2).
   a. Do not change the attribute SCRIPT_FILE or any DESCR_<Nr.> attribute (<Nr.> is 1...9).

b. Enter for PARAM_1 the value for the minute (1-60), at which the Central Backup will be started. It is possible to give different start minutes delimited by comma or * for all minutes.
   E.g. in MAS_EXP for start up at the first minute of an hour: 1
   For more information, see the manpages of cron and crontab in appendix.

c. Enter for PARAM_2 the value for the hour (1-24), at which the Central Backup will be started.
   It is possible to give different start hours delimited by comma or * for all hours.
   E.g. in MAS_EXP for start up to 10 p.m. (at the twentysecond hour): 22
   For more information, see the manpages of cron and crontab in appendix.

d. Enter for PARAM_3 the value for day of the month (1-31), at which the Central Backup will be started.
   It is possible to give different start days of the month delimited by comma or * for non day (then only day of the week will be used, see item f.).
   E.g. in MAS_EXP for start day of the month at 1st, 7st, 14st and 21st of the month: 1,7,14,21
   For more information, see the manpages of cron and crontab in appendix.

e. Enter for PARAM_4 the value for month (1-12), at which the Central Backup will be started.
   It is possible to give different start month delimited by comma or * for all month.
   E.g. in MAS_EXP for start up every month: *
   For more information, see the manpages of cron and crontab in appendix.

f. Enter for PARAM_5 the value for day of the week (0-6), at which the Central Backup will be started.
   It is possible to give different start days of the week delimited by comma or * for non day (then only day of the month will be used, see item d.).
   E.g. in MAS_EXP for start day of the week every Friday: 5
   For more information, see the manpages of cron and crontab in appendix.

NOTE

In the MAS_EXP project example the Time controlled Central Backup will be started every 1th, 7th, 14th and 21th and additional every Friday on each month at 10:01 p.m. The time base is the AS500ES date and time, given from HP-UX system. The target date and time system will not be considered or used.

g. Enter for PARAM_6 the value for the first network, which will be used by Central Backup.
   It is only allowed to give one network number.
   E.g. in MAS_EXP for network 11(MB300): 11
   For more information, see in ONB UserManual and AS500ES UserManual.

h. Enter for PARAM_7 the value for the target/node number of the first network, which will be saved by Central Backup.
   To save more than one target/nodes in first network, the node numbers must be delimited by space.
   E.g. in MAS_EXP for nodes 12, 13, 14 and 24 in first network: 12 13 14 24
   For more information, see in ONB UserManual and AS500ES UserManual.

i. Enter for PARAM_8 the value for the second network number for the wanted Central Backup.
   It is only allowed to give one network number.
   E.g. in MAS_EXP for network 12(MB300): 12
   For more information, see in ONB UserManual and AS500ES UserManual.

j. Enter for PARAM_9 the value for the target/node number of the second network, which will be saved by Central Backup.
   To save more than one target/node in second network, the node numbers must be delimited by space.
   E.g. in MAS_EXP for nodes 22, 23, 24 and 34 in second network: 22 23 24 34
   For more information, see in ONB UserManual and AS500ES UserManual.

NOTE

In the MAS_EXP project example the nodes 12, 13, 14 and 24 of network 11 and the nodes 22, 23, 24 of network 12 will be saved by the Time controlled Central Backup.

4. Accept your values and exit TEB.
5. Open Object “START_CRON” again.
6. Select “FREE” in appearing “Open Object” window and press “OK” button, now the Time controlled Central Backup will be activated and the corresponding entries will be made in crontab of the HP-UX system.

NOTE

The Time controlled Central Backup files will be saved in the same way as with the do-restore script (Tool of ONB), please see in Usermanual for ONB and Central Backup for further information

7. Check in the information file, which will be displayed with vuepad now, that you are not in conflict with other user, which already started eventually the Time controlled Central Backup, too. In the case of a conflict, stop the wrong Central Backup, see Chapter F.1.1.1 Stop of an already running Time controlled Central Backup.
8. For check of correct running of Time controlled Central Backup see logfile /tmp/time_cent_back.log with the log of the Central Backups and settings of Time controlled Central Backup.

WARNING

By wrong use and not allowed syntax/values no error checks exists in Time controlled Central Backup. Only the normal Tool-checks of ONB - Central Backup and HP-UX system will be made!

F.1.1.3 Restore of Time controlled Central Backup files

The saved files can be restored with the do_restore script (Tool of ONB) in the same way as for the normal Central Backup, please see in User Manual for ONB and Central Backup for further information.

F.1.2 Installation of Time controlled Central Backup in other Projects

1. Copy the object "FREE_LIB" with its children objects "START_CRON" and "STOP_CRON" by use of <FILE>+<Import...> menu under, "PROJECT_LIBRARY" in your new project.
2. Open a hpterm window
3. Change to directory /proj/<proj_name>/util/free/ e.g.:
   ```
   cd /proj/stst_h1/util/free/
   ```
   (<proj_name> is the project name of item 1.)
4. Save your original freeopen.sh file :
   ```
   cp freeopen.sh freeopen.sh.org
   ```
5. Copy the new freeopen.sh file from MAS_EXP project :
   ```
   cp /proj/mas_exp/util/free/freeopen.sh .
   ```
6. Copy the startcron.sh and stopcron.sh file from MAS_EXP project into the new project:
   ```
   cp /proj/mas_exp/util/free/startcron.sh startcron.sh
   cp /proj/mas_exp/util/free/stopcron.sh stopcron.sh
   ```
7. Change the correct user access for the new files:
   ```
   chmod 774 freeopen.sh startcron.sh stopcron.sh
   ```

F.1.3 Working Method of Time controlled Central Backup

F.1.3.1 STOP_CRON

The FREE object "STOP_CRON" will start first freeopen.sh script as described in the Usermanual for FREE object.
The new freeopen.sh script will then call script stopcron.sh with the parameter from FREE object (in this case none).
The script stopcron.sh removes a user's crontab in HP-UX system and so the Time controlled Central Backup for this user is stopped.

F.1.3.2 START_CRON

The FREE object "START_CRON" will start first freeopen.sh script as described in the Usermanual for FREE object.
The new freeopen.sh script will then call the script stopcron.sh with the parameter of the FREE object.
The script startcron.sh will interpret the parameters. One part for the Central Backup and the other part for the crontab.
For the Central Backup a script startdobackup will be created including the call of do_backup (Tool of ONB). Please see in User Manual for ONB and Central Backup for further information regarding do_backup script (e.g. restrictions of do_backup will also be valid for Time controlled Central Backup).
For crontab an entry will be made in the HP-UX system with the start parameter (date, time and script startdobackup for start Central Backup).
As information to avoid conflicts with other crontab users a file with information regarding the current crontab settings will be created and displayed.

F.1.4 Appendix for Time controlled Central Backup

F.1.4.1 Manual Page of crontab

crontab(1) crontab(1)

NAME

crontab - user crontab file

SYNOPSIS

crontab [file]

crontab -r

crontab -l

DESCRIPTION

crontab copies the specified file, or standard input if no file is specified, into a directory that holds all users' crontab files (see cron(1M)). The -r option removes a user's crontab from the crontab directory. crontab -l lists the crontab file for the invoking user.

Users are permitted to use crontab if their names appear in the file /usr/lib/cron/cron.allow. If that file does not exist, the file /usr/lib/cron/cron.deny is checked to determine if the user should be denied access to crontab. If neither file exists, only root is allowed to submit a job. If only cron.deny exists and is empty, global usage is permitted. The allow/deny files consist of one user name per line.

A crontab file consists of lines of six fields each. The fields are separated by spaces or tabs. The first five are integer patterns that specify the following:

minute (0-59),
hour (0-23),

day of the month (1-31),
month of the year (1-12),
day of the week (0-6 with 0=Sunday).

Each of these patterns can be either an asterisk (meaning all legal values), or a list of elements separated by commas. An element is either a number, or two numbers separated by a hyphen (meaning an inclusive range). Note that the specification of days can be made by two fields (day of the month and day of the week). If both are specified as a list of elements, both are adhered to. For example, 0 0 1,15 * 1 runs a command on the first and fifteenth of each month, as well as on every Monday. To specify days by only one field, the other field should be set to * (for example, 0 0 * * 1 runs a command only on Mondays).

The sixth field of a line in a crontab file is a string that is executed by the shell at the specified times. A percent character in this field (unless escaped by \) is translated to a new-line character. Only the first line (up to a % or end of line) of the command field is executed by the shell. The other lines are made available to the command as standard input.

Hewlett-Packard Company HP-UX Release 9.0: August 1992

NOTE

You can also get this information by command: `man crontab`
For further information use command man <keyword>.

F.1.4.2 Manual Page of cron

cron(1M) cron(1M)

NAME

cron - clock daemon

SYNOPSIS

/etc/cron

DESCRIPTION

cron executes commands at specified dates and times. Regularly scheduled commands can be specified according to instructions found in crontab files. Users can also submit their own crontab file via the crontab(1) command. Commands that are to be executed only once can be submitted by using the at command. Since cron never exits, it should be executed only once. This is best done by running cron from the initialization process through the file /etc/rc (see init(1M)).

cron only examines crontab files and at(1) command files during process initialization and when a file changes. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

In the HP Clustered environment, the directories /usr/lib/cron and /usr/spool/cron are context dependent files (CDFs). Each cnode is expected to run its own copy of cron.

NOTE

On the days of daylight savings (summer) time transition (in time zones and countries where daylight savings time applies), cron schedules commands differently than normal.

In the following description, an ambiguous time refers to an hour and minute that occurs twice in the same day because of a daylight savings time transition (usually on a day during the Autumn season). A non-existent time refers to an hour and minute that does not occur because of a daylight savings time transition (usually on a day during the

Spring season). DST-shift refers to the offset that is applied to standard time to result in daylight savings time. This is normally one hour, but can be any combination of hours and minutes up to 23 hours and 59 minutes (see tztab(4)).

When a command is specified to run at an ambiguous time, the command is executed only once at the first occurrence of the ambiguous time.

When a command is specified to run a non-existent time, the command is executed after the specified time by an amount of time equal to the DST-shift. When such an adjustment would conflict with another time specified to run the command, the command is run only once rather than running the command twice at the same time.

For commands that are scheduled to run during all hours by specifying a * in the hour field of the crontab entry, the command is scheduled without any adjustment.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If LANG is not specified or is set to the empty string, a default of "C" (see lang(5)) is used instead of LANG. If any internationalization variable contains an invalid setting, cron behaves as if all internationalization variables are set to "C". See environ(5).

EXAMPLES

The following examples assume that the time zone is MST7MDT. In this time zone the DST transition occurs one second before 2:00 a.m. and the DST-shift is 1 hour.

Consider the following crontab entries:

```
# Minute   Hour    Month Day   Month   Weekday   Command
________________________________________________________
  0       01        *         *       *     Job_1
  0       02        *         *       *     Job_2
  0       03        *         *       *     Job_3
  0       04        *         *       *     Job_4
  0        *        *         *       *     Job_hourly
  0       2,3,4     *         *       *     Multiple_1
  0       2,4       *         *       *     Multiple_2
```

For the period of 01:00 a.m. to 04:00 a.m. on the days of DST transition, the results will be:

Job	Times Run in Fall	Times Run in Spring
Job_1	01:00 MDT	01:00 MST
Job_2	02:00 MDT	03:00 MDT
Job_3	03:00 MST	03:00 MDT
Job_4	04:00 MST	04:00 MDT
Job_hourly	01:00 MDT	01:00 MST
	02:00 MDT	
	02:00 MST	
	03:00 MST	03:00 MDT
	04:00 MST	04:00 MDT
Multiple_1	02:00 MDT	
	03:00 MST	03:00 MDT
	04:00 MST	04:00 MDT
Multiple_2	02:00 MDT	03:00 MDT
	04:00 MST	04:00 MDT

WARNINGS

In the Spring, when there is a non-existent hour because of daylight savings time, a command that is scheduled to run multiple times during the non-existent hour will only be run once. For example, a command scheduled to run at 2:00 and 2:30 a.m. in the MST7MDT time zone will only run at 3:00 a.m. The command that was scheduled at 2:30 a.m. will not be run at all, instead of running at 3:30 a.m.

DIAGNOSTICS

A history of all actions taken by cron is recorded in /usr/lib/cron/log.

AUTHOR

cron was developed by AT&T and HP.

FILES

/usr/lib/cron	main cron directory
/usr/spool/cron	spool area
/usr/lib/cron/log	accounting information

SEE ALSO

at(1), crontab(1), sh(1), init(1M), cdf(4), queuedefs(4), tztab(4).

STANDARDS CONFORMANCE

cron: SVID2

The shell is invoked from your $HOME directory with an initial argument of sh. Users who desire to have their .profile executed must explicitly do so in the crontab file. cron supplies a default environment for every shell, defining HOME, LOGNAME, SHELL=/bin/sh, and PATH=:/bin:/usr/bin.

EXTERNAL INFLUENCES

International Code Set Support

Single-byte character code sets are supported.

WARNINGS

Be sure to redirect the standard output and standard error from commands. If this is not done, any generated output or errors are mailed to the user.

FILES

/usr/lib/cron	main cron directory
/usr/lib/cron/cron.allow	list of allowed users
/usr/lib/cron/cron.deny	list of denied users
/usr/spool/cron/crontabs	directory containing the crontab files
/usr/lib/cron/log	accounting information

SEE ALSO

sh(1), cron(1M), queuedefs(4).

STANDARDS CONFORMANCE

crontab: SVID2, XPG2, XPG3

Hewlett-Packard Company HP-UX Release 9.0: August 1992

NOTE

You can also get this information by command: `man cron`
For further information use command man <keyword>.

INDEX

A

AAX file 3-285
ABB templates 1-28
Accelerator keys 1-34
Accept/Save 1-36, 3-87
Act 3-108
Activation flag 3-108
Administration 3-52, 3-60
  Exit 3-188
Administration Logfiles 3-56
Administration Program Actions 3-59
Administration Window 3-60
Administrator
  Password 3-201
  User Name 3-201
AdvantController 400C 3-282
AdvantStation 500ES 3-282
AdvantStation 500IMS 3-282
AdvantStation 500OS 3-282
all linktypes 3-221
Application Program 3-47
AS 100ES directory structure 3-265, 3-278
AutoCAD 1-23

B

Backtranslate 3-49, 3-285
Backup 3-293
BAS 3-109
Basic Functions 1-9
BAX file 3-285
Build the Project Structure 3-76
Builders 1-8
button 1-33

C

CAD 1-23
CAPE 1-8
CFG-Password 3-62
child 1-7
click 1-29
close a subtree 3-7
COM 3-109
command line options 3-77
Comment 3-205
comment 3-6
Copy from project 3-204
Copy only single object 3-204, 3-207
Copy to project 3-204
Corrupted Project 3-72
Create a New Project 3-60
Create Links 3-96
Create New Objects 3-81

D

Data exchange with Advant Station 100 3-263
  via Floppy Disk 3-266
  via Network 3-268
db start 2-1
DBDATA file 3-285
Delete a Project 3-71
Delete a User 3-75
Delete Project dialog box 3-71
Delete User dialog box 3-76
DET 3-109
Diagram Builder 1-23
dialog box 1-33
DISP 3-83
Display Builder 1-26
display element 1-26
DOB_Configure window 3-108
DOB_Generation window 3-102
DOC_RELATION 3-100
Document Builder 1-28
document package 3-99
Document packages 3-20
Document Structure 3-112
Document structure 3-20
DOCUMENT TYPE 3-113
double-click 1-29
Down-load 3-43
drag 1-29
DSTRUC 3-20
DSTRUC object 3-101

E

Edit Object Attributes 3-88
EIU 3-159
Electrical Diagram Builder 1-24
Enlarge a Project 3-70
Enlarge Project Size dialog box 3-70
Error Messages 3-82
Error messages
  Maintenance 5-45

es_shutdown 2-2
Export 3-172
  ASCII file 3-159
Export of AS 100ES Files 3-275
Export/Import functions for AS 100ES files 3-265
Export/Import Utility 3-159

F

field 1-33
File conversion routines 3-280
Filter
  Query 3-223
filter
  list dialog 3-80
Filter object id 3-214
Filter object type 3-214
FORMS 1-29
FORMS user interface 1-34
FrameMaker 1-28
free project space 3-70
Function Chart Builder 1-20

H

Help 3-82
  index 3-188, 3-227
  on Context 3-188
  On Keys 3-188
  On Version 3-188, 3-227
Help Window 3-82
HP VUE 1-29
HP-UX Log In dialog box 3-1

I

I&C 3-109
Id filter 3-212
Import 3-163
  ASCII file 3-159
  AutoCAD file 3-181
  FrameMaker file 3-181
Import of AS 100ES Files 3-269
Import/Export utility 1-23
INS 3-109
Insert a User 3-63
Insert Mode 3-88
Insert Object 3-12
Insert Objects 3-89
Insert Project 3-61
Insert User 3-64
IS_DOCUMENTATION_OF 3-91, 3-113
IS_DOCUMENTATION_OF link 3-101
IS_IMPLEMENTED_BY 3-91
Items 3-215, 3-219

L

label 1-33
Link 3-92
  copy 3-96
  create 3-96
  General Link Handling Window 3-95
  General Link Handling window 3-92
  Handling 3-91
  Link Type Specific window 3-92 to 3-93
  LINK_TYPE 3-95
  OBJ_ID1 3-91, 3-93, 3-95
  OBJ_ID2 3-91, 3-93, 3-95
  OBJ_TYPE1 3-95
  OBJ_TYPE2 3-95
  Of Next Level 3-222
  Of Object 3-221
  Of Whole Subtree 3-223
  PAR_OBJ_ID1 3-97
  query 3-97
  TYPE_CODE 3-93
Link Attributes 3-94
List 1-35
  object 3-212
List button 3-79
List Function 3-88
List of Values 3-10, 3-88
list presentation 1-12, 3-79
Loan Flag 3-84
LOAN_FLAG 3-84
Log In 3-1
Logfile
  Administration 3-56
logfile 3-56

M

Mask mode 3-204, 3-207, 3-210
Mask new comments 3-208, 3-211
Mask new ids 3-208, 3-210
MasterBatchController 200 3-282
MasterBus 300 3-282
MasterGate 230 3-282
MasterPiece 200 3-282
MasterPiece 51 3-293
MasterView 800 3-282
Matched ids 3-215

maximize 1-31
menu 1-34
message line 1-37
minimize 1-31
MNT 3-109
MOD_EXP 3-5
Modify a User's Password 3-74
Modify Project dialog box 3-69
Modify User dialog box 3-75
Mouse 1-29
multiple object form 3-13

N
Navigate
    list presentation 3-80
    tree presentation 3-78
Navigate in the List Presentation 3-80
Net, Node number 3-283
New parent id 3-204
Node Name 3-283

O
OBJ_ID 3-84
Object 3-83, 3-212
    comment 3-205
    copy 3-81, 3-206
    delete 3-211
    deleting 3-82
    ID 3-205
    Identification 3-84
    identifier 1-9
    import 3-81
    Information 3-83, 3-218
    insert 3-81, 3-205
    locking 3-87
    move 3-208
    moving 3-82
    open 3-216
    Open Type 3-217
    print 3-217
    Print Type 3-218
    rename 3-209
    renaming 3-82
    type 3-206
object 1-7
Object form
    multiple 1-18, 3-85
    single 1-18, 3-84
Object ID 3-6
Object id 3-204 to 3-205, 3-224
Object Information Window 3-83
Object Information window 3-18
Object Type 3-6, 3-206
Object type 3-224
Old object comment 3-208, 3-210
Old object id 3-207, 3-210
On same level 3-204, 3-206 to 3-207, 3-209
On same level button 3-12
On-line Builder 1-22, 3-282
On-line Builder Convert Dialog window 3-283
ON-LINE view 3-283
only open links 3-221
Open a Project 3-77
ORACLE SQL*Loader 3-159

P
parent 1-7
Parent button 3-8
parent button 3-78
Parent objec 3-224
Password 3-201
    Edit 3-200
    modify 3-74
PC element 3-288
PC item designation 3-288
PC program 3-288
PC source code file 3-288
PCDATA files 3-283
PCPGM object 3-283
PDB Configure Plant Documentation 3-99
PDB Generate Plant Documentation 3-99
PDB_Configure window 3-100
PDB_Generation window 3-22
PLA 3-109
Plant Documentation Builder 1-22, 3-99
pointer 1-29
poweroff 2-2
Predecessor 3-207, 3-209
print only to file 3-202, 3-218
printing documentation packages 3-99
Project
    class 3-190
    comment 3-62
    corrupted 3-72
    delete 3-71, 3-196
    description 3-62, 3-190
    enlarge 3-70, 3-196
    Free space 3-190

  - freeze 3-65
  - insert 3-61, 3-194
  - modify 3-69, 3-197
  - Name 3-62
  - name 3-190
  - number 3-190
  - Print 3-191
  - protection 3-190
  - release 3-65, 3-190
  - restore 3-67, 3-193
  - save 3-66, 3-192
  - Size 3-62
  - size 3-190
  - source 3-63, 3-190
  - standard 3-60
  - Start-Date 3-190
  - Status 3-190
  - type 3-190
- Project Administration 3-60
- Project name 3-204
- project number 3-64
- Project object 3-6
- Project Root 3-20
- project root 3-215
- project size 3-70
- Project status 3-58
- project user 3-64
- project's comment 3-69
- project's protection 3-69
- Projects Window 3-53
- Projects window 3-61, 3-189
- Pro-Size 3-70
- Protect Project 3-63

Q

- Query 3-182
  - cancel 3-184, 3-187
  - Count Hits 3-187
  - count hits 3-183
  - enter 3-182, 3-185
  - execute 3-184, 3-187
- query
  - list dialog 3-80
- Query - Enter 3-185
- Query - Execute 3-187
- Query Count Hits 3-187
- Query Menu 3-185

R

- RDR-Password 3-62
- reader user 3-63
- RealTime Accelerator Board 3-282
- Rename Objects 3-82
- Rename only single object 3-210
- Replace mode 3-88
- Reports
  - SELECT Statement 3-132
- resize 1-31
- Restore 3-294
- Restore a Project 3-67
- Restore Project dialog box 3-68
- root 1-7, 3-6
- root object 3-79

S

- Save a Project Release 3-65
- Save Project dialog box 3-66
- scroll bar 1-32
- Search list 3-214
- Search tree 3-214
- Search tree and list 3-214
- Selected link type 3-221
- Selected view 3-218
- Selection 3-204, 3-206, 3-215, 3-219
- Set New View Directed dialog box 3-97
- Set New View Undirected dialog box 3-97
- show description 3-221
- Show Keys 1-35
- shuffle 1-31
- Single-object form 3-9
- SOLUTION 3-143
- Solution Aspects window 3-144, 3-229
- SOLUTION object 3-143
- Solution State 3-145
- Solution State button 3-144
- SOLUTIONPOOL object 3-143
- Source code 3-285
- Source Code File 3-41
- Special commands 3-219
- special commands 3-219
- SQL* ReportWriter 3-120
- SQL*ReportWriter 1-22
- SRCE directory 3-288
- status line 1-37
- STRUC object 3-101
- Structure
  - Control System 1-10

- Function 1-10
- Location 1-10
- Plant Documentation 1-10

Structure Builder 1-9, 3-76
SUB_ID 3-109
Substitute comment 3-208, 3-211
Substitute id 3-207, 3-210
Substitute mode 3-204, 3-207, 3-210
subtree 1-7
subtree button 3-6, 3-78

T

Target Node 3-42
TBX file 3-285
Template Builder 1-18, 3-9, 3-84
temporary root 3-8, 3-79, 3-213
TIX file 3-285
toggle button 1-33
tree 1-7
tree presentation 1-11
Type filter 3-212

U

Up-load 3-49
User
- ADMIN 3-199
- CFG 3-62
- CONFIG 3-199
- configuration 3-63
- delete 3-75, 3-200
- group 3-62
- Group No. 3-199
- insert 3-63, 3-199
- name 3-199
- password 3-199
- Print 3-198
- RDR 3-62
- READER 3-199
- WRITER 3-199
- writer 3-63

User name 3-201
user's password 3-74
Users Window 3-53
Users window 3-63, 3-197

V

Views 3-218

W

Whole subtree 3-212, 3-224
window 1-30
writer 3-63

3BSE 001 972R0201

April, 1997